General Disclaimer

One or more of the Following Statements may affect this Document

- This document has been reproduced from the best copy furnished by the organizational source. It is being released in the interest of making available as much information as possible.

- This document may contain data, which exceeds the sheet parameters. It was furnished in this condition by the organizational source and is the best copy available.

- This document may contain tone-on-tone or color graphs, charts and/or pictures, which have been reproduced in black and white.

- This document is paginated as submitted by the original source.

- Portions of this document are not fully legible due to the historical nature of some of the material. However, it is the best reproduction available from the original submission.

Produced by the NASA Center for Aerospace Information (CASI)

OU TM-47 (NASA)

NASA CR- 145178

A MICROCOMPUTER-BASED LOW-COST OMEGA SENSOR PROCESSOR

(NASA-CR-145178) A MICROCOMPUTER-BASED LOW-COST OMEGA SENSOR PROCESSOR M.S. Thesis (Ohio Univ.) 146 p HC A07/MF A01 CSCL 17G

N77-22063

Unclas
G3/04 25116

by

Richard J. Salter, Jr.

Avionics Engineering Center
Department of Electrical Engineering
Ohio University
Athens, Ohio 45701

February 1977

Supported by

National Aeronautics and Space Administration

Grant NGR 36-009-017

MAY 1977
RECEIVED
NASA STI FACILITY
INPUT BRANCH

FOREWORD

This paper was submitted to and approved by the Department of Electrical Engineering and the Graduate College of Ohio University on March 4, 1977 as a master's thesis.

ABSTRACT

A microcomputer-based, low-cost Omega sensor processor (OSP) has been designed, implemented with minimum hardware, and tested. The feasibility of implementing a minimum hardware/maximum software system of less complexity and lower cost than is currently available has been verified. The design approach and hardware/software details are presented for this low-cost navigation aid.

The long-range project goal of developing a general aviation Omega navigation system (ONS) for a cost under $1000 is now feasible. The microcomputer-based OSP presented here represents nearly all the system hardware needed for an airborne ONS. An area navigation routine is the additional component needed to complete the ONS for general aviation. This navigation routine can readily be added to the present system since it is additional software and not hardware.

This paper describes the OSP design and engineering approach used in achieving this design. In reflecting a structured program design approach, the project goal of designing a low-cost OSP is presented initially. Succeeding chapters document the efforts made in support of this goal. The OSP operations are functionally described in the text, as is the system hardware and software. Circuit details of the hardware modules and program listings for the FORTRAN and microcomputer software are reserved for the appendices.

TABLE OF CONTENTS

LIST OF FIGURES

I. INTRODUCTION

This project was sponsored by the NASA/Langley Research Center under a grant effort known as the NASA Tri-University Program in Air Transportation Systems. The program was begun in 1971 and includes MIT, Ohio University, and Princeton University in a cooperative effort to improve the safety, reliability, and efficiency of our National Airspace System. Each University offers a unique area of specialized interest and expertise, thereby complementing each others' efforts; Ohio University's area of specialization being avionics.

Early in the program's development it was determined that the Tri-University Program might make a contribution to general aviation by developing a low-cost area navigation system for small aircraft. Studies pointed to the Omega navigation system as a low-cost system which could provide all-altitude signal coverage, and the major emphasis in the program became the development of a low-cost Omega navigation system for the general aviation pilot.

While MIT studied the systems integration aspects and Princeton University the flight dynamics aspects, Ohio University turned its efforts towards designing a more sensitive and lower cost Omega receiver than was commercially available. An early result was an Omega sensor processor incorporating all TTL circuitry, and this unit was flight-tested aboard the Avionics Engineering Center's DC-3 flying laboratory enroute to Langley Field, Virginia in October, 1974. After this successful flight demonstration, Ohio University embarked on a project to build five Omega prototype receivers for the other Universities and NASA. These prototypes were designed using low power CMOS circuitry and were delivered to the recipients in September, 1976.

The Omega sensor processor outputs, Omega phase and phase differences, are the inputs needed by a navigation processor to derive pilot-usable information such as cross-track error and down-track distance. Due to the recent availability of low-cost microcomputers, efforts were channeled toward using a commercially-available microcomputer as the navigation processor for the airborne Omega navigation system (ONS). When incorporating a microcomputer into the ONS, however, it became obvious that, in addition to performing the area navigation functions, the sensor processor functions could also be performed in microcomputer software.

Therefore, a project was begun to explore the feasibility of a minimum hardware/maximum software OSP. This paper reports results of this project: a minimum hardware system has been designed, and the OSP functions of automatic synchronization, phase tracking, and line-of-position (LOP) generation have been implemented in microcomputer software. The microcomputer-based OSP thus consists of a 10.2 KHz TRF receiver module, an interface module which samples the receiver outputs, and a microcomputer module to process the Omega phase data.

The design approach was to first develop the system hardware (receiver and interface modules) and use these modules to supply Omega data to a digital magnetic tape recorder. A data base of real Omega ground and flight data was collected for use in a FORTRAN language simulation analysis of OSP functions. Algorithms to perform the OSP functions were developed and optimized via this simulation. The routines were then assembled into microcomputer code and loaded into the OSP microcomputer module. With the receiver and interface supplying

Omega data to the microcomputer module, the output data consistently equaled or exceeded the quality of the all-hardware OSP outputs, thus verifying the feasibility of the software-based system.

The hardware and software for the microcomputer-based low-cost OSP are functionally described herein, and the type of outputs available and the results obtained are presented. The hardware circuit details and software routines program listings are documented in the appendices.

II. OMEGA SENSOR PROCESSOR

The worldwide Omega navigation system is a network of eight VLF transmitting stations. The stations transmit in a time-multiplexed fashion on the frequencies of 10.2 KHz, 13.6 KHz, and 11.333 KHz. A brief system description is given in Appendix A, while a more thorough treatment is afforded by U.S. Coast Guard publication "Omega Navigation System User Handbook". [1] In-depth treatment of Omega system details has been given by Pierce [2] and Morris. [3]

Information on the design of airborne receiving and navigating systems using the Omega signals is less accessible. Although commercial manufacturers are understandably secretive with their receiver hardware/software designs, reports on those systems whose development has been funded by government agencies should be available to the public (e.g., " Omega Navigation Set: An/ARN-99 Final Engineering Report" [4]). An all-hardware Omega sensor processor designed by Ohio University for NASA is described in "Ohio University Omega Prototype Receiver". [5]

An Omega navigation system for airborne use can consist of three functional units: an Omega sensor processor (OSP), navigation processor, and a cockpit display. The OSP receives the Omega signals and provides amplification and signal-to-noise ratio improvement. The OSP must provide Omega data usable by the navigation processor. This navigation processor must convert the Omega information into pilot-usable information to be displayed on the cockpit display. A simple display sufficient for general aviation could take the form of analog or digital readout of cross-track error and down-track distance.

A. Design Approach. The project effort to develop a low-cost Omega navigation system for general aviation has been greatly aided by the recent commercial availability of inexpensive microcomputers. The reasons for chosing to use a microcomputer to perform navigation processing are well known: high reliability, small size, low cost, low power consumption, and off-the-shelf availability in addition to software design flexibility. Using a microcomputer for sensor processing as well as navigation processing has the advantages noted above, but it can present some complex systems problems to the designer (particularly time sharing of the computer for several functions and prioritizing of work). Therefore, this project was undertaken to explore the feasibility of implementing an Omega sensor processor with a minimum hardware/maximum software combination.

The design approach taken was to use the previously developed Ohio University low-cost front-end receiver module [6] in conjunction with an interface module to collect a data base of real Omega ground and flight data. This data base was then used as the input for a FORTRAN language simulation analysis

of the OSP functions. After the OSP functions had been simulated and optimized in FORTRAN, they were rewritten in microcomputer language and tested in the final system configuration. The power of this approach lies in the fact that the software routines were developed not only with computer-generated data as input, but they were optimized to perform with the actual Omega data as collected with the OSP hardware.

The OSP is functionally illustrated in Figure 1, and the hardware modules and software algorithms are described in the following two sections. For the purpose of data collection, the microcomputer module of Figure 1 was replaced with a digital magnetic tape recorder. This setup was used in the laboratory for the collection of ground-based data, while the collection of flight data was accomplished with this configuration onboard Ohio University's DC-3 aircraft N4002. Complete documentation of the Omega phase data base is given in Appendix B.

Figure 1 shows that the basic system operation is as follows: the receiver front-end module receives the Omega signal from the antenna-preamplifier and provides filtering, amplification, and hard-limiting before sending Omega zero-crossings to the interface module. The microcomputer interface module measures and samples the Omega phase information at some predetermined rate and generates a microcomputer interrupt signal. The microcomputer (or data recorder) receives the digitized phase sample, and this information is used by the OSP software routines.

The OSP software routines for automatic synchronization, phase tracking, and LOP generation perform the bulk of the Omega phase processing work.

B. System Hardware. The Omega sensor processor design is intended to be a minimum hardware configuration with emphasis on low cost. The use of a general purpose microcomputer to perform as many OSP functions as possible reflects this design objective. The OSP is functionally illustrated in the block diagram of Figure 1, and a description for each of the hardware modules follows. Circuit details for each module are given in Appendix C.

1. Receiver. The antenna used for airborne operation is the aircraft's ADF sense antenna. If the aircraft is not equipped with an ADF sense wire, another high-impedance E-field antenna may be used (such as a bent dipole nav-com antenna). For ground-based operation it is desirable to mount the antenna as high above buildings, etc., as possible and in an area free of 60 Hz interference (10.2 KHz being the 170th harmonic of 60 Hz).

The signal developed across the antenna is amplified by a preamplifier which is colocated with the antenna. This colocation is necessary to overcome signal attenuation due to capacitance of the receiver module lead-in cable. The preamplifier is primarily an impedance-matching device and has two outputs. The Omega output provides a gain of 20 decibels at 10.2 KHz in a -3 dB bandwidth of several kilohertz. The phase shift is adjusted to zero degrees at 10.2 KHz. The ADF output provides a flat response of -6 dB across the band from 10 KHz to 1 MHz, and the phase shift is zero degrees across the ADF band. The preamplifier response characteristics are shown in Figure 2. The preamplifier has been described by Wright.[7]

Accepting the signal from the preamplifier is a 10.2 KHz TRF receiver front-end module (see Figure 3). The front-end module provides additional gain,

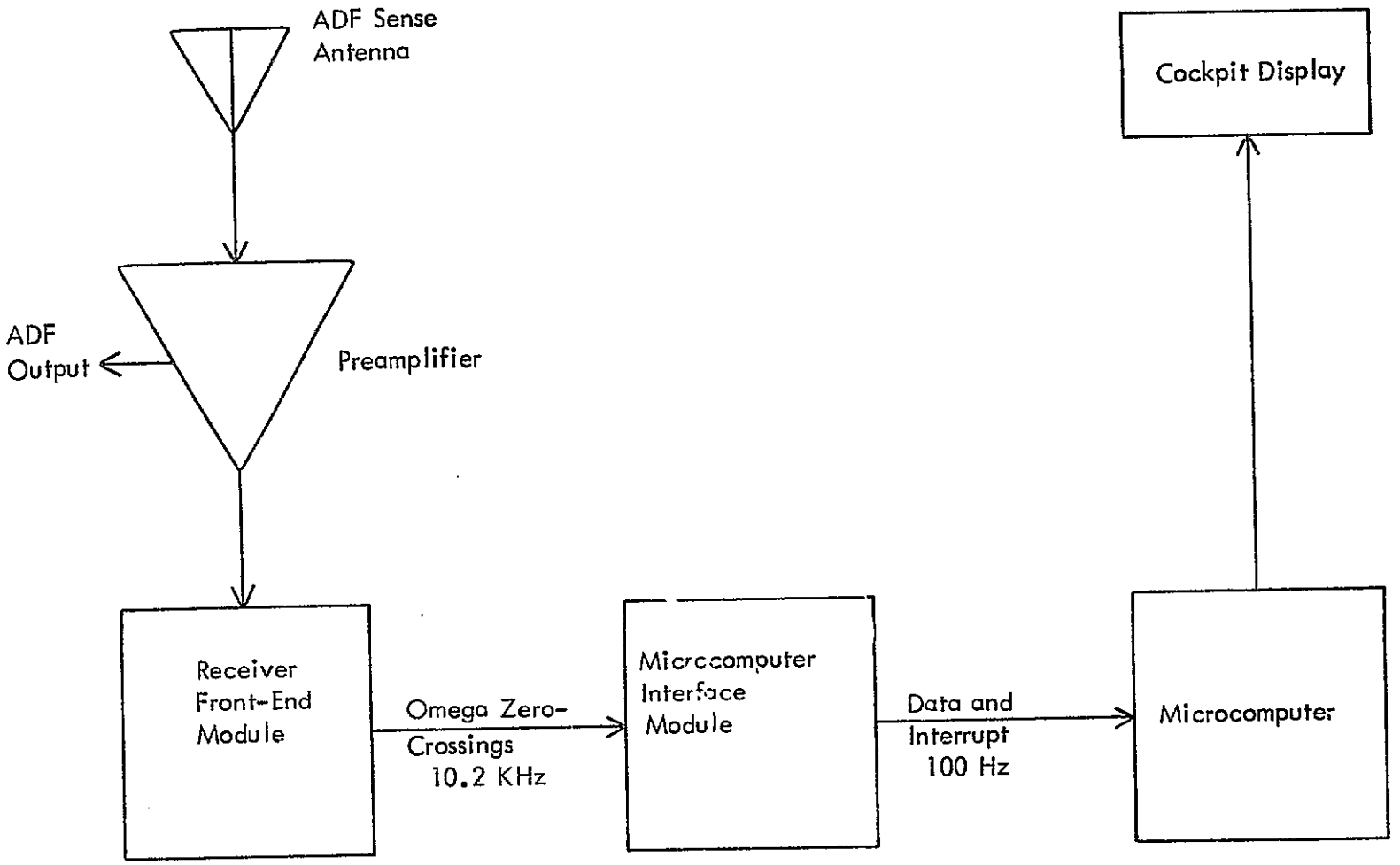


Figure 1. Summary Block Diagram - Microcomputer-Based Omega Sensor Processor.

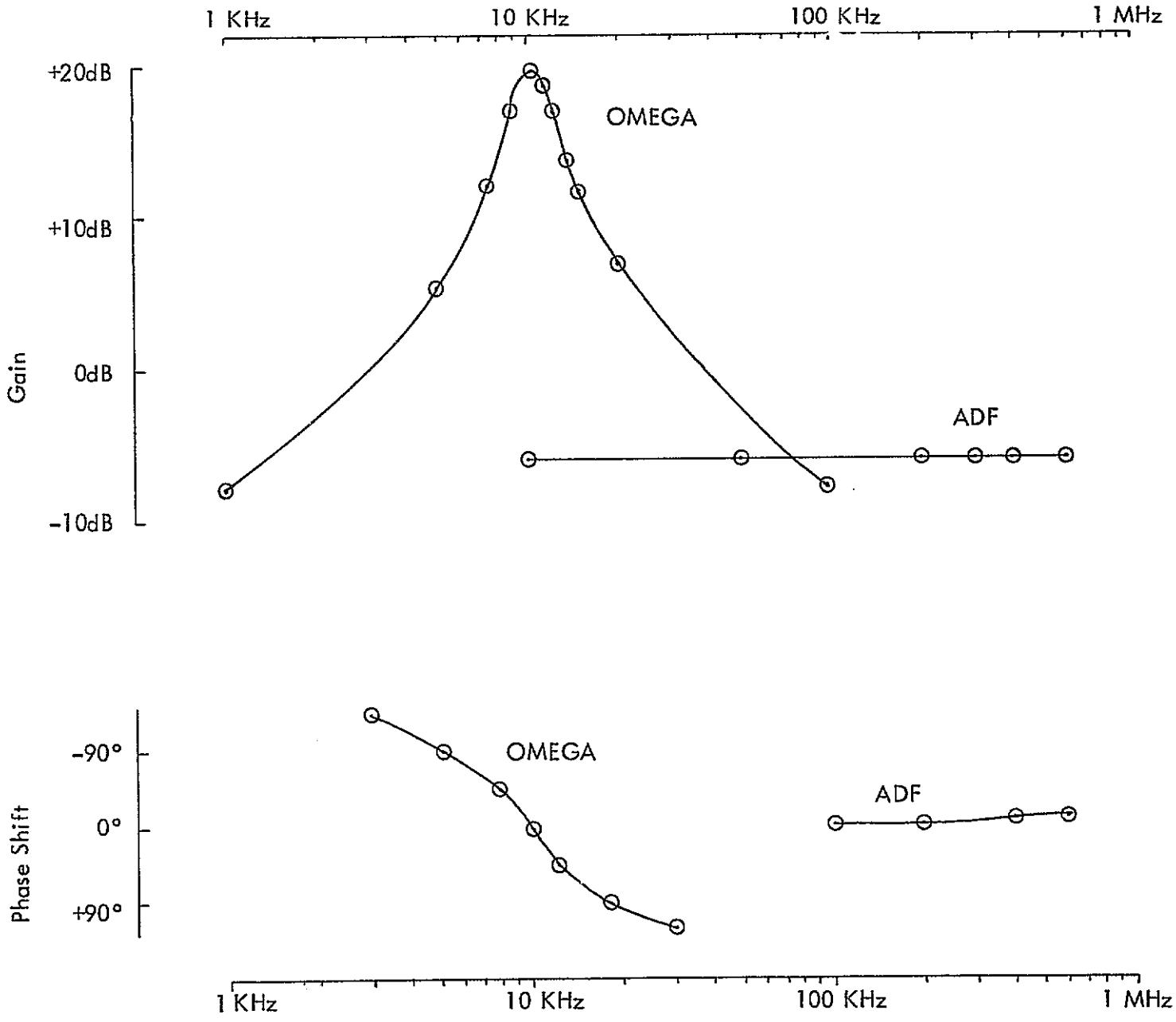


Figure 2. Preamplifier Characteristics.

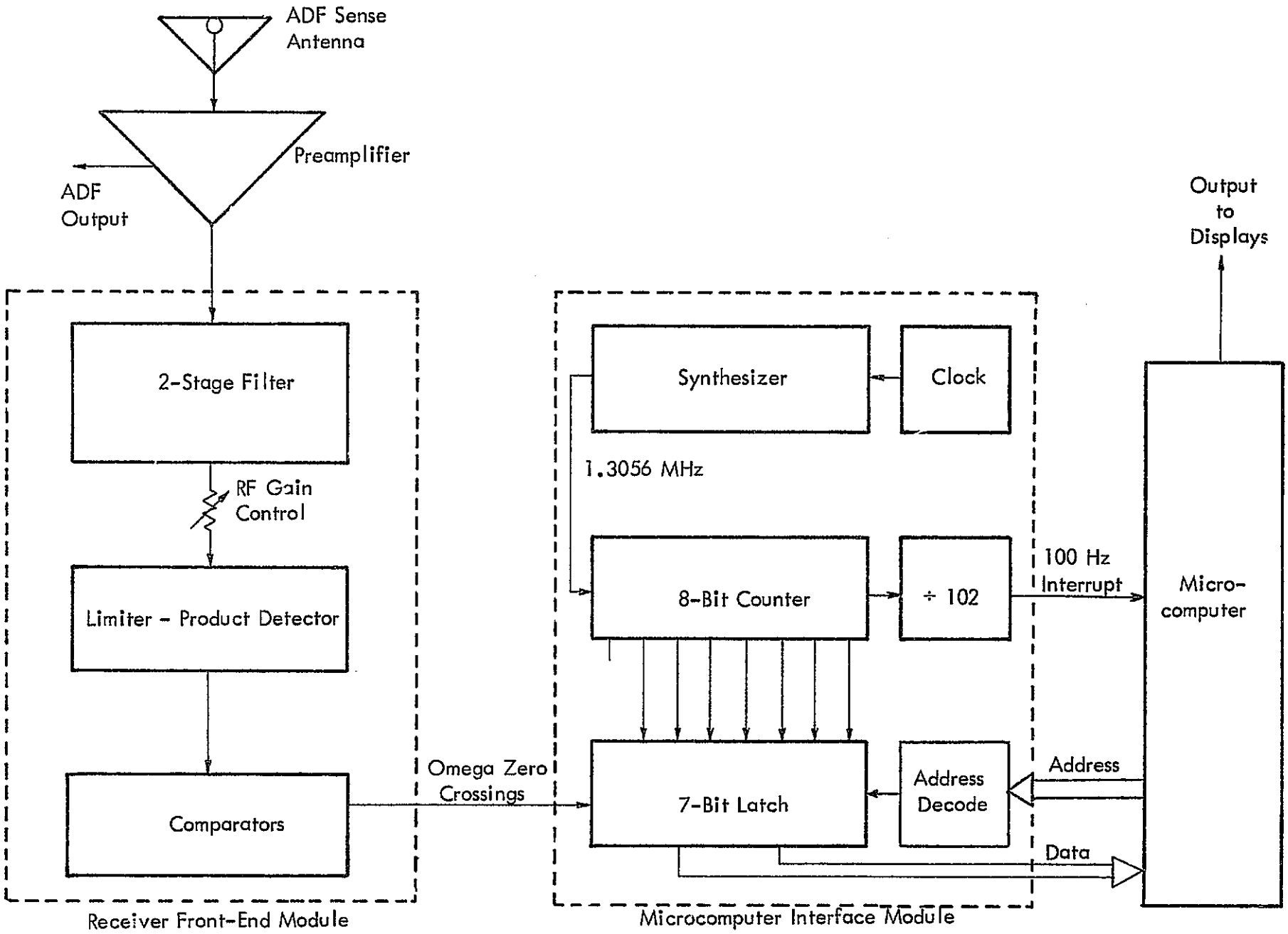


Figure 3. Front-End, Computer Interface, and Microcomputer Modules Block Diagram.

narrowband filtering, and hard-limiting (i.e. provides pulse outputs coincident with the 10.2 KHz Omega zero-crossings). The unit consists of three integrated circuit chips and two ceramic filters providing a bandwidth of 30 Hz at 10.2 KHz. The module is equipped with an RF gain control which permits gain adjustment from 160 to 200 dB. The front-end frequency response curve is shown in Figure 4. As can be seen from this curve, the response is down nearly 80 dB at the other Omega frequencies of 11.333 KHz and 13.6 KHz. The sensitivity and recommended operating range is 1 microvolt to 10,000 microvolts. The front-end module has been described by Burhans.[6]

2. Computer Interface. After the bandpass filtering and hard-limiting provided by the front-end, pulses coincident with the Omega zero-crossings are sent to the microcomputer interface module. This module is also illustrated in Figure 3. The functions of phase detection and data sampling are performed by this interface hardware, and it also generates the interrupt signals necessary for reading the phase data into the microcomputer. Besides the Omega zero-crossings, a clock signal of 1.3056 MHz is used by the microcomputer interface module. This clock signal is synthesized from a 5 MHz temperature compensated crystal oscillator. The choice of 5 MHz as the basic oscillator frequency permits operation by an external, more stable clock of this standard frequency should this be desired.

Phase detection is performed in an open loop fashion: the 2^7 x 10.2 KHz clock signal is input to a 7-bit reference counter, and the seven counter outputs are used as data inputs to a 7-bit latch. The Omega zero-crossings clock the latch, thus providing a relative phase measurement of Omega compared to the local clock approximately 10,200 times per second.

ORIGINAL PAGE IS
OF POOR QUALITY

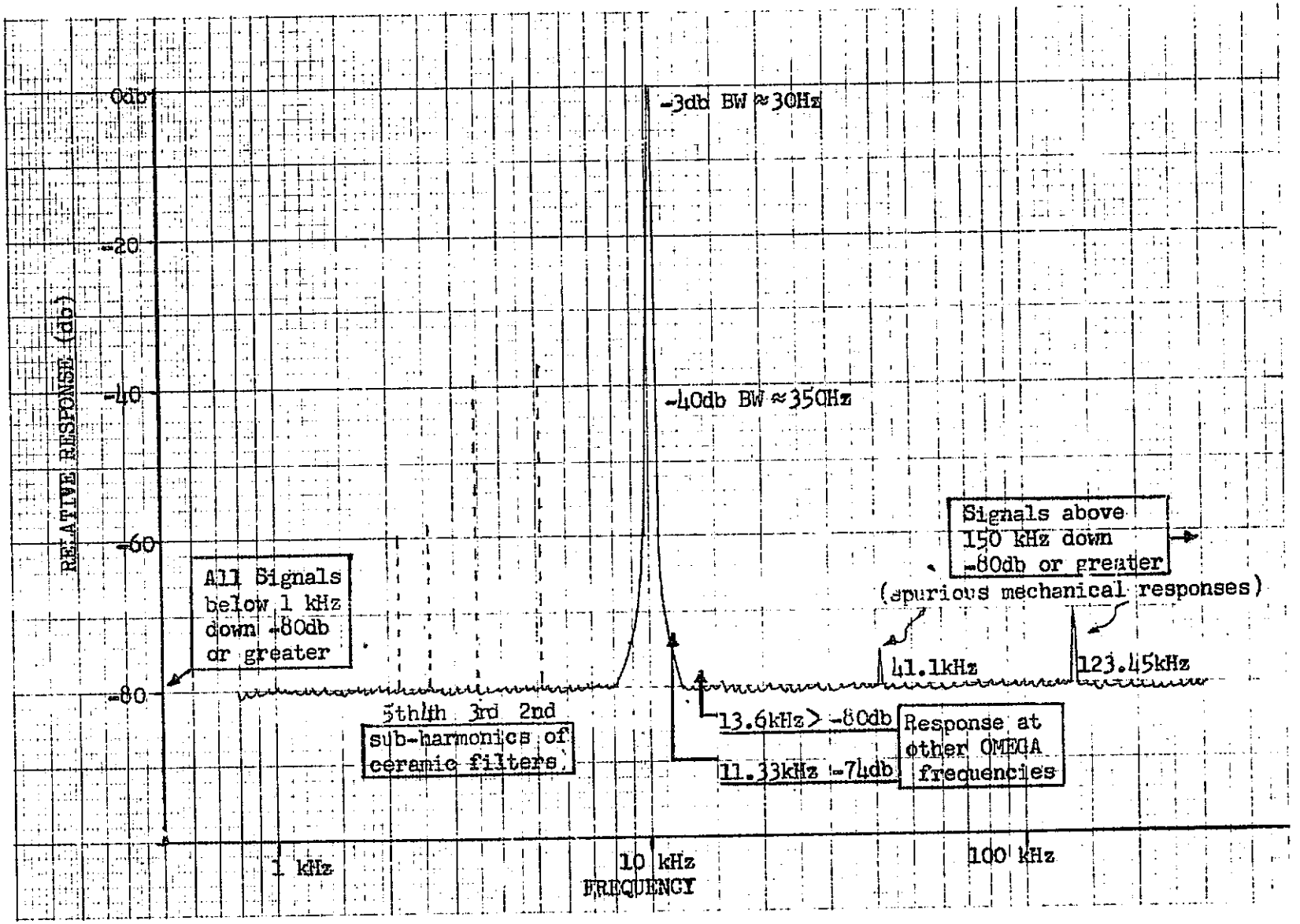


Figure 4. Front-End Module Response Curve.

There are at least two considerations which suggest sampling the Omega phase measurements at a rate slower than the 10.2 KHz carrier rate:

(1) Since an extremely precise measurement is not required, it is unnecessary to use the phase measurement of every cycle of the carrier to gain a filtered estimate of the true phase.

(2) Using the data at the maximum available raw-data rate implies a prohibitively high interrupt rate for most microcomputers.

The choice of a particular sampling rate is also based on these considerations. In this implementation a sampling rate of 100 Hz has been chosen for the following reasons:

(1) Sampling of the 15 Hz post-detection bandwidth phase at 100 Hz is a rate high enough to avoid severe aliasing of the sampled data spectrum, and it provides a sufficient statistical base of data points for each Omega time-slot.

(2) 100 Hz is an acceptable microcomputer interrupt rate, and it is easily generated from the existing timing circuitry.

The microcomputer interface module derives the 100 Hz sampling frequency, samples the phase data at this rate, and sends an interrupt signal to the microcomputer. Having received the interrupt signal, the microcomputer services the interrupt request by loading the new phase data from the interface module. The inexpensive technique known as memory-mapping is used to accomplish this loading of data. The microcomputer address bus is decoded, thereby enabling

the data onto the data bus and into the microcomputer (as indicated in Figure 3). The interface module and memory-mapping have been previously addressed by this author in references 8,9, and 10.

3. Microcomputer. The microcomputer module consists of a commercially available microcomputer system made up of a central processing unit (CPU), random access memory (RAM), peripheral interface adapter (PIA), teletypewriter interface circuitry, and teletypewriter monitor software in permanent read-only storage (ROM) as illustrated in Figure 5. The CPU utilizes a sixteen-line address bus, enabling addressing up to 65 kilobytes of memory. An eight-bit bidirectional data bus allows for data transfer among the CPU, memory, and input/output (I/O) devices.

The computer's clock is a 1 MHz adjustable RC oscillator or crystal oscillator, resulting in a machine cycle time of one microsecond. Most instructions require 2 or 3 machine cycles to execute. CPU registers include one accumulator, two index (scratch-pad) registers, a processor status register, a stack pointer, and a program counter. All of these are eight-bit registers except the program counter which is sixteen bits. Appendix C gives the microcomputer specifics, and its operation is explained in detail in the manufacturer's manuals.[11,12]

Figure 6 shows the simple, inexpensive method for output of the microcomputer data, which is the same technique used for data input mentioned previously. An output latch is memory-mapped to a particular address by decoding the microcomputer address bus lines. The address decoder output signal is gated with the microcomputer's "write" signal. This gate's output enables a latch which stores the logic levels appearing on the data bus at that

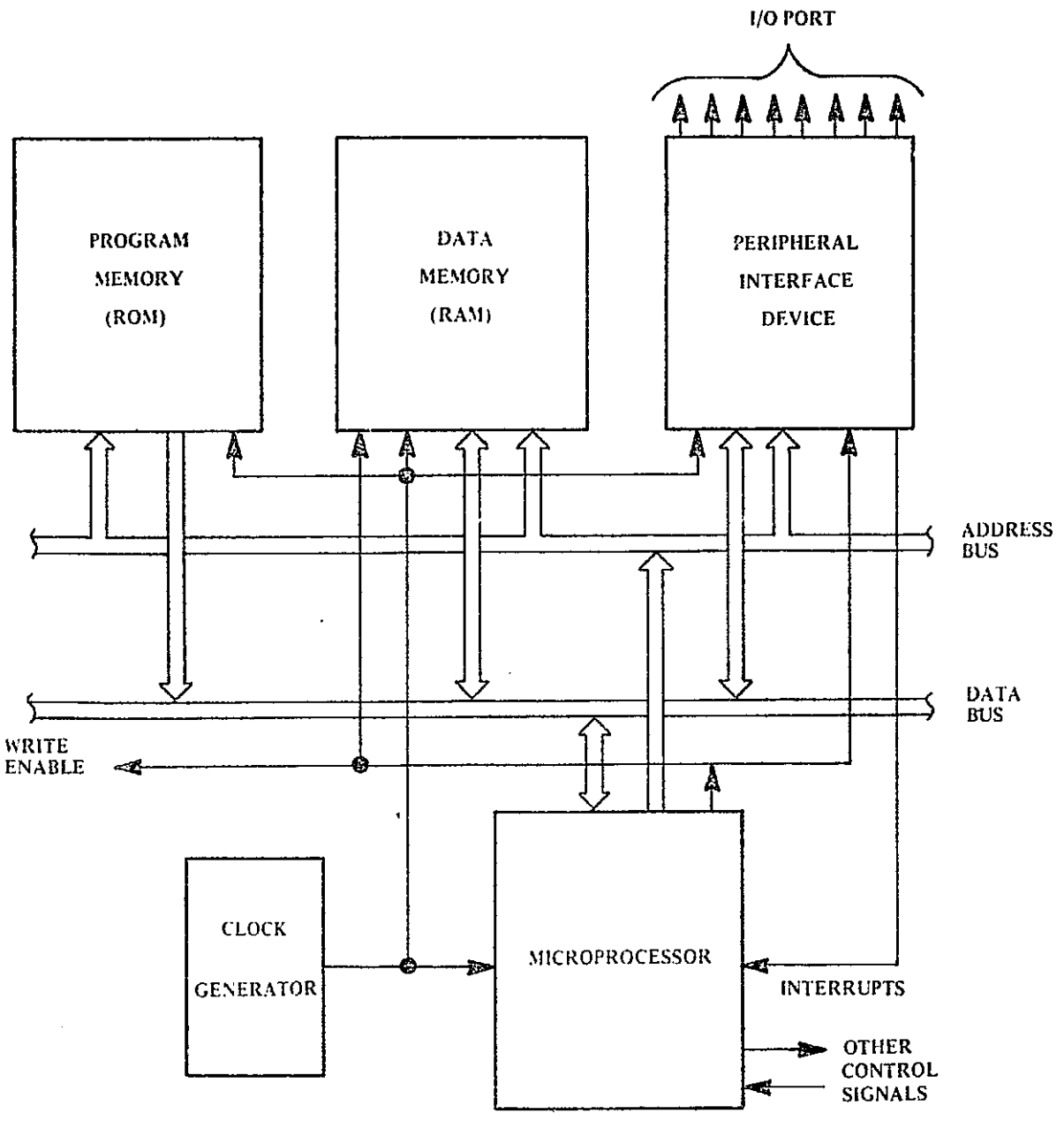


Figure 5. Microcomputer System Organization.

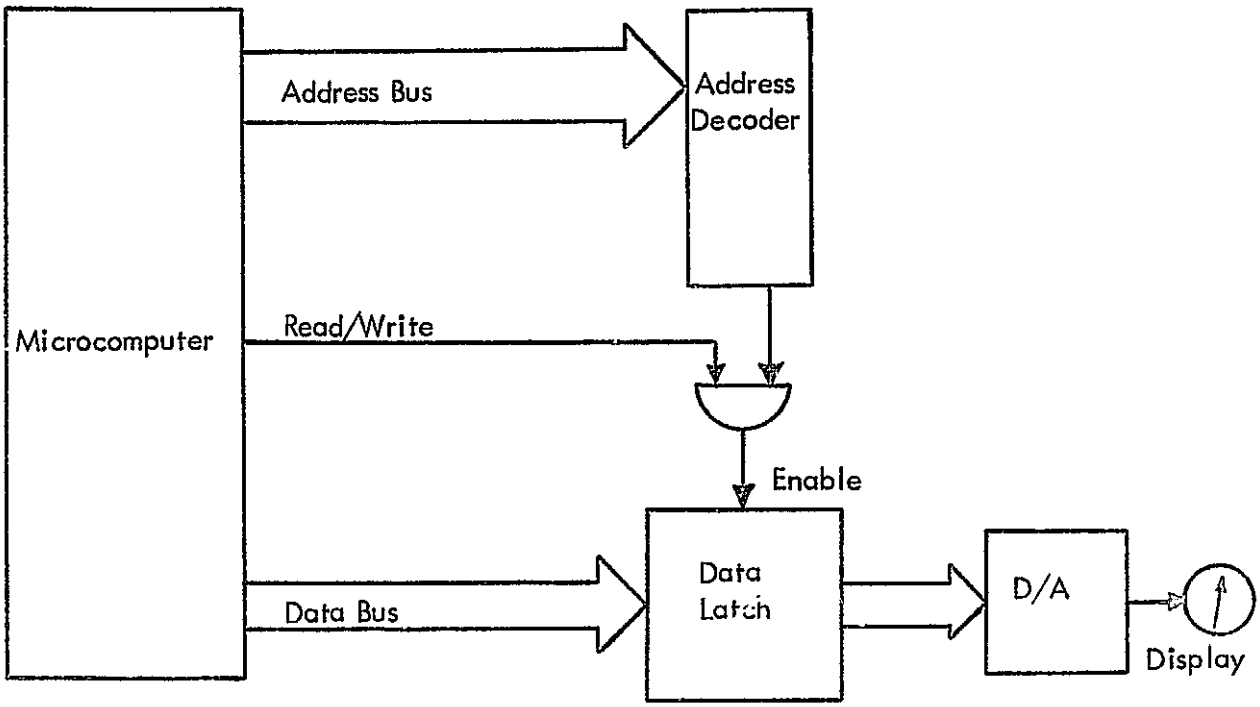


Figure 6. Memory-Mapped Data Output Port.

time. This data is then fed to a digital-to-analog (D/A) converter for display on a chart recorder or meter.

C. System Software.

1. Interrupt Servicing. When the microcomputer interface module sends an interrupt signal to the microcomputer, the interrupt service routine loads the new data into the computer. Then the software process as flow-charted in Figure 7 begins. The basic interrupt service routine serves as a system time-keeper and data "traffic cop": it loads in new data as it becomes available and hands it off to the proper data-processing subroutine. If the OSP is not in sync with the Omega transmission format, synchronization must be achieved first. Otherwise, the data is sent to the tracking loop subroutine if it is time-slot data, whereas the data is simply ignored if it is from an Omega transmission gap. A data output subroutine is executed if it is the correct time for it.

The automatic synchronization, tracking loop, and data output subroutines called by this interrupt service routine are functionally described in the subsequent text material, while the FORTRAN and microcomputer language programs are listed and described in Appendices E and F. The interrupt service routine requires 150 bytes of memory and performs its time and data management in a few tens of microseconds following each interrupt request signal.

2. Automatic Synchronization. The Omega system consists of eight stations transmitting on three VLF channels in a fixed, time-multiplexed format. As described in Appendix A, a unique pattern is formed by the scheduled length (in time) of each station's transmission which repeats every ten seconds. Thus, to use the phase measurements supplied by the microcomputer interface module,

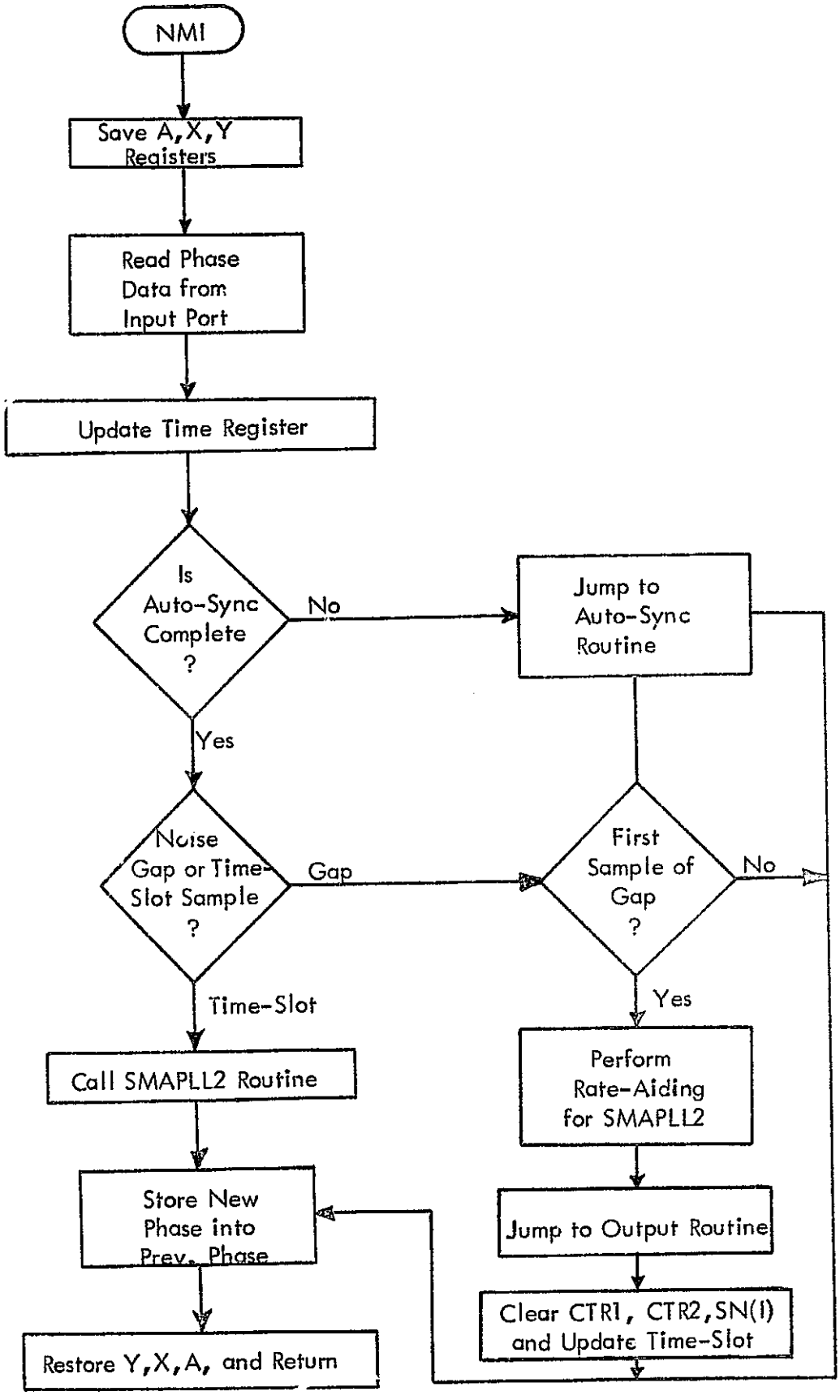


Figure 7. Interrupt Service Routine Flow Chart.

some means must be provided for synchronizing the OSP timing with the Omega transmission pattern. Only after the OSP is "in sync" can the phase of each station's signal be followed by tracking filters to give position-fixing information.

It should be pointed out that precision sync (better than plus or minus 100 msec) is unnecessary. A .2 second gap between each transmission burst allows for propagation delays from transmitter to receiver and for signal build-up on the transmitting antenna and through the narrowband receiver module without signal overlap into another station's time-slot. Therefore, sync within plus or minus 100 msec insures that only one station will be received during each of the eight time-slots.

A description of the automatic synchronization procedure simulated in FORTRAN and implemented in the OSP microcomputer follows, and a logic flow diagram appears in Figure 8. Although the routine is described here in terms of a single-frequency OSP, the technique is applicable to multiple-frequency systems as well. Performing the routine on multiple channels simultaneously would give the OSP a redundancy check. By averaging the results obtained for all channels a best estimate of the true sync point could be obtained.

To establish sync a pattern of station transmission lengths is derived from the received data, and the entire frame of data is shifted until the received pattern coincides with the unique transmission pattern stored in the computer. The received pattern of transmission lengths can be established by determining the amount of coherence between consecutive 10 msec phase samples. A high level of coherence indicates the presence of a strong signal, while the lack of phase coherence (i.e. phase jitter) is indicative of a weak signal or noise. The absolute value of phase

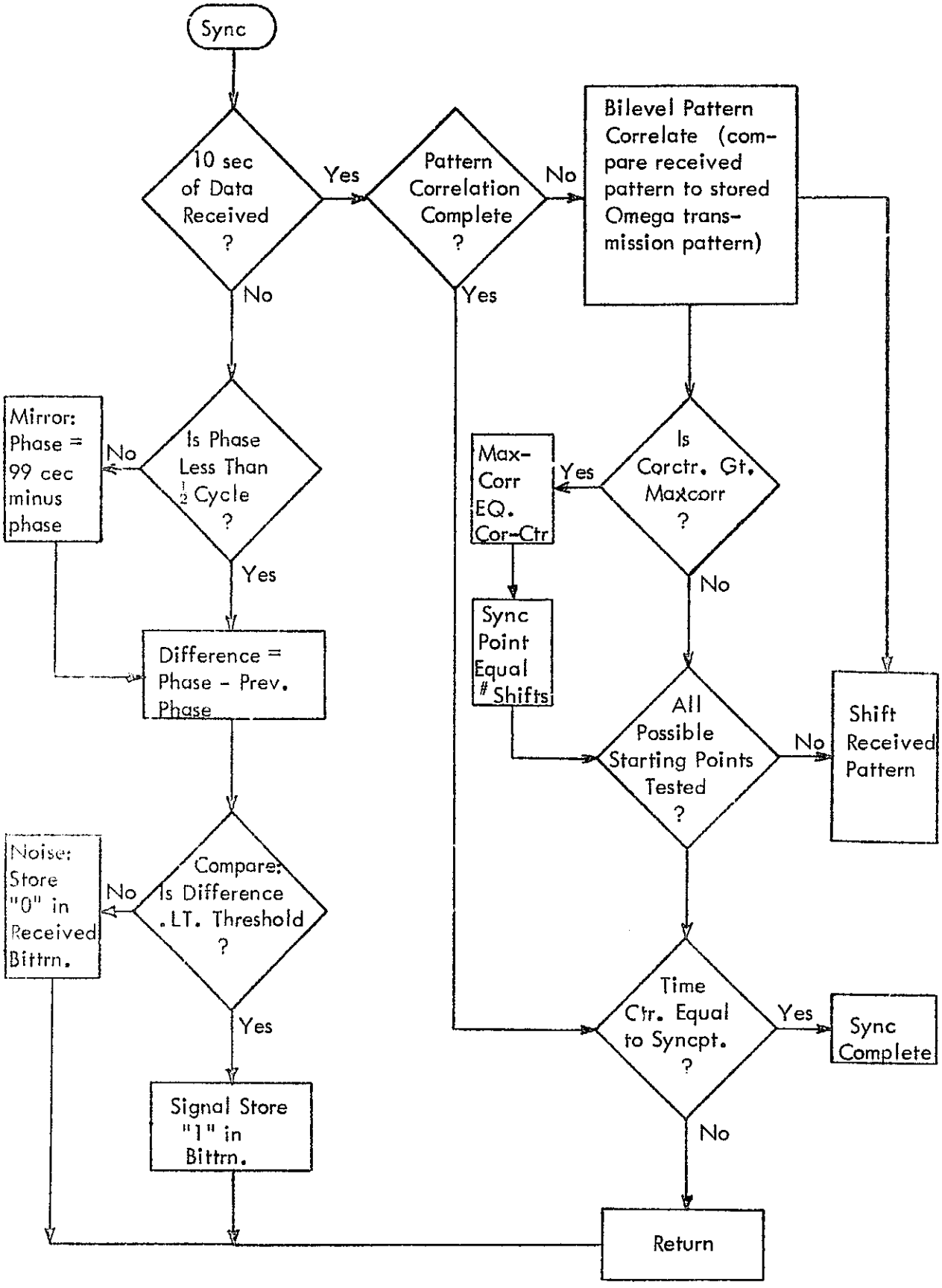


Figure 8. Automatic Synchronization Routine Flow Chart.

measurements is meaningless during the sync process; only the phase coherence from sample to sample is useful in determining the received transmission pattern (i.e. in determining whether each sample is signal or noise).

An Omega phase measurement is supplied by the microcomputer interface module every 10 msec (100 Hz sampling rate). Starting at a random point in time (power on time) and taking data for ten seconds (one complete Omega frame) yields 1000 phase samples. This is all the information needed to accomplish sync. The Omega signal is contaminated with noise, causing a series of measurements to result in a dispersion of phase points rather than a constant value. The units of centicycles (cec) are commonly used in Omega jargon. One cec is 1/100 of an Omega carrier cycle; at 10.2 KHz a cycle is about 16 nautical miles long, making one cec equal to about .16 miles. If the phase of the received signal with respect to the OSP clock is near the 0 or 99 cec measurement extremes (near the edge of a cycle), the measured value may "bobble" between these extremes. To remove this bobble each data point above 49 cec is "mirrored" about the 50% full-scale value (mirrored about 49 cec). Points at x = 99 cec are thus reflected to 99-x = 0 cec. Since only the phase coherence is of interest and not the absolute value of phase, the mirroring process merely puts the data into a better form from which coherence can be determined.

A simple routine yielding a measure of the point-to-point coherence is to take the difference of each two successive phase points. That is, subtracting each measurement from the previous yields 1000 difference values for the 10 second data frame. Each difference is inversely proportional to the coherence between the two phase samples.

By comparing each difference value to a predetermined "threshold constant", a decision can be made as to whether signal or noise exists in each 10 msec sample slot. However, it is desirable to first smooth or average the difference values over several sample slots before comparison to the threshold constant is made. If the smoothed difference is less than the threshold, a "1" is stored for the sample slot; whereas, if the smoothed difference is greater than the threshold, a "0" is stored for the slot (low difference ⟶ high coherence ⟶ signal present). After all 1000 comparisons have been performed, a pattern of 1000 1's and 0's represents the signal and noise sample slots, respectively. The "high-low" or "on-off" pattern created by the 1000 binits resembles the Omega transmission format shown in Appendix A, the amount of resemblance being proportional to each station's signal-to-noise ratio (SNR).

Note that the 1000 bits can be stored in 125 eight-bit bytes of memory (an amount of storage not prohibitive for simple microcomputer systems). Also bear in mind that the entire microcomputer will be dedicated to the auto-sync routine during the sync process, since no Omega navigation information is available until sync is completed.

The 1000 sample bits are next compared to the stored Omega pattern in a bilevel correlation process. Use is made of the time-slot and transmission gap information; whenever the received slot binit is equal to the stored slot binit, a correlation counter is incremented by one. After all 1000 slots have been compared, the stored pattern is shifted by one slot and the correlation of points sequence is repeated. This shift and correlate iteration is continued until all 1000 possible shifted patterns have been tested. The number of shifts necessary

to obtain the highest bilevel correlation value is used to obtain the starting point for the "A" time-slot (assuming the microcomputer's interrupt service routine has been keeping time (counting 1000 Hz interrupts) in a modulo 10-second fashion since the auto-sync process began).

Having found the sync-point in the simulation data, Figure 9 shows the original 1000 byte record of phase data plotted ten points per line. The higher the SNR of the station being received in each time-slot, the less dispersion there is in the phase points plotted.

The OSP operator's a priori knowledge of which stations are on the air and which ones will be well received in his particular geographic area can be incorporated into the creation of the unique transmission pattern stored in the computer. This facilitates the bilevel correlation process, resulting in a more accurate sync point.

Although it was not used in this version of the OSP, for an additional confidence measure the entire sync process can be performed several times and a best estimate of the sync point computed. If multiple frequency channels are available, the process can be performed on each channel and the sync points averaged (or a sync point can be considered acceptable only if it results on two of three channels, etc.). Another possibility is to save the second highest correlation value as well as the highest, and consider the sync point acceptable only if the highest value is better than the second highest by some predetermined amount.

In summary, the technique presented requires very simple microcomputer instructions (compare, subtract, shift), uses very little scratch-pad RAM storage

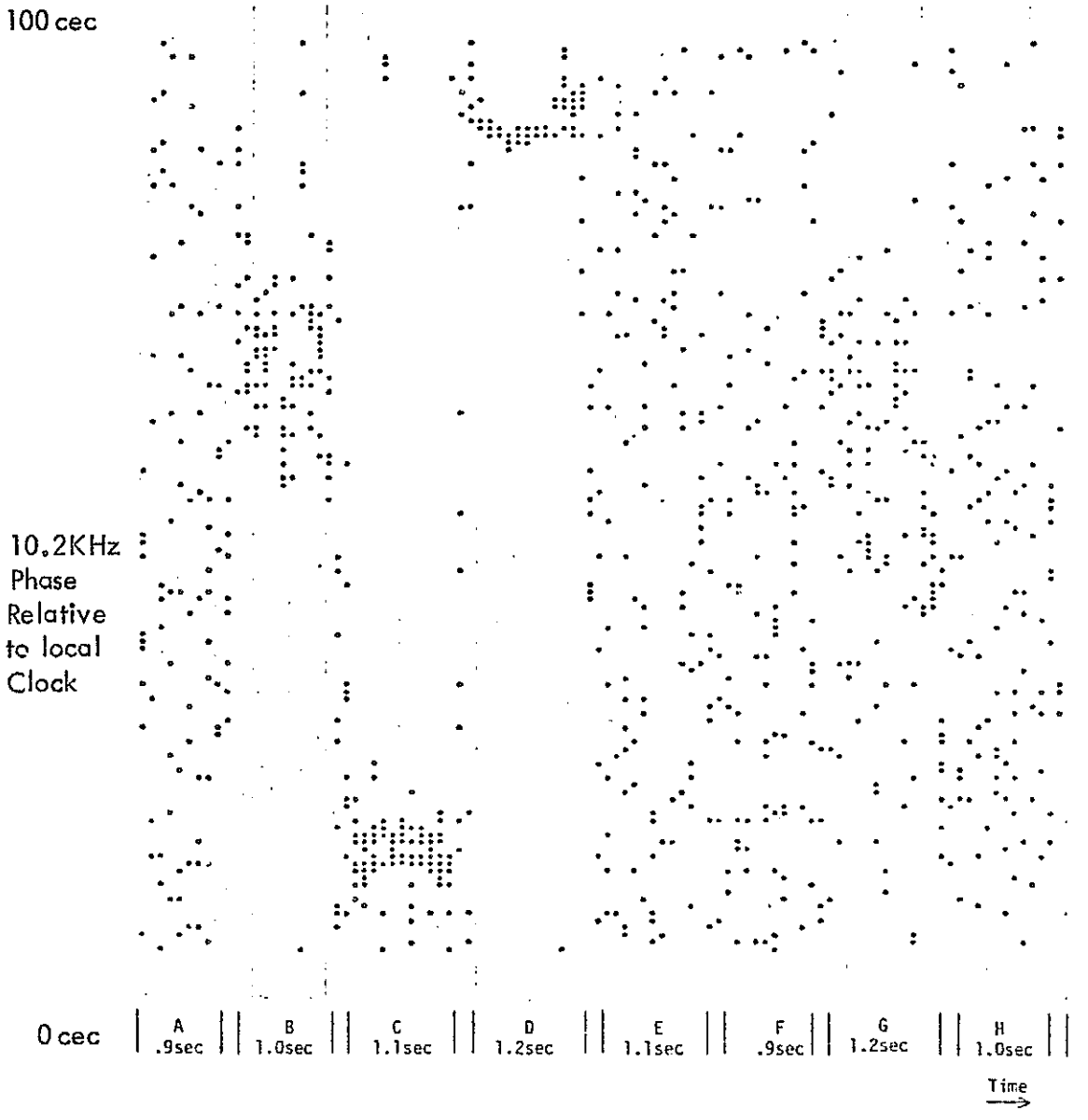


Figure 9. Computer Plot of Raw Omega Data. This is the 30 Hz bandwidth phase plotted 10 points per line.

(125 bytes), and usually accomplishes sync in about 100 seconds (depending on the redundancy constraints imposed). The technique has demonstrated acceptable accuracy (within ± 100 msec of true sync) and the microcomputer program is about 300 bytes long.

3. PLL Phase Tracking. After the OSP is in sync with the Omega transmissions, the phase of each station's signal can be tracked using correlation techniques, and SNR improvement can be made. The correlation detection structure known as the memory-aided phase locked loop (MAPLL) has been found to be a successful tracking filter for the Omega signals.[13] Since previous experience with hardware versions of the MAPLL had revealed its usefulness for Omega tracking, it was desirable to duplicate this configuration in software.

The first-order version of the software memory-aided phase locked loop (SMAPLL1) is functionally illustrated in Figure 10. The loop operation can be generally explained as follows: the loop's bilevel quantizing phase detector compares the incoming phase measurement with a filtered loop control word (LCW) at the 100 Hz sampling rate. The output of the phase detector is an increment command if the incoming phase is ahead of the LCW; it is a decrement command if the incoming phase lags behind the LCW. The increment/decrement commands are filtered in a bidirectional loop counter. Loop filtering is achieved by requiring the loop counter to receive several correlated signals (counts in the same direction) before it affects the LCW. When the loop counter has received sufficient correlated up/down commands, it outputs an increment/decrement command to the LCW register. This new LCW is in turn compared to the new incoming phase measurements in the phase detector.

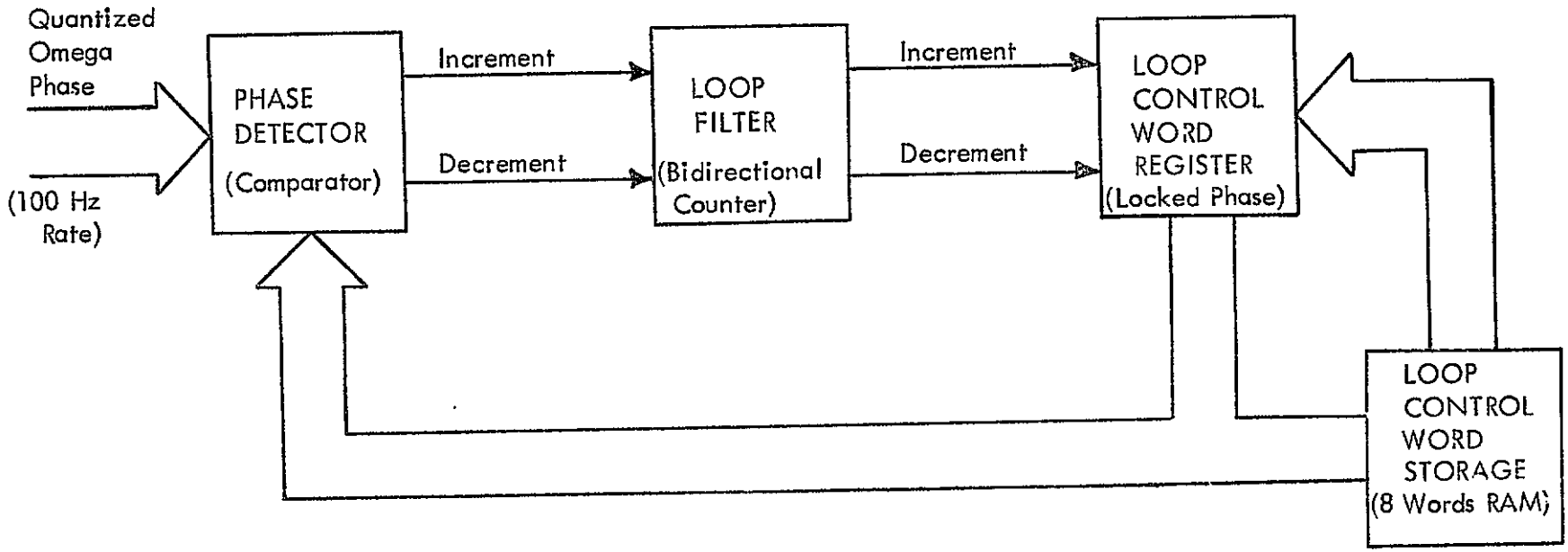


Figure 10. First-Order Software Memory-Aided Phase Locked Loop (SMAPLL1) Block Diagram.

The memory-aiding feature allows a single phase locked loop (PLL) routine to be utilized to track all eight Omega stations' phase. At the end of a current time-slot the LCW is stored in a memory location, and the LCW for the next time-slot is retrieved from memory and used to track the new time-slot's phase.

Atmospheric noise in the VLF-Omega frequency range is known to be caused primarily by lightning.[14] The noise is made up of both gaussian and impulsive noise. The bilevel quantizing phase detector is thought to discriminate against the impulsive noise better than a linear detector[13] (in addition to being simpler to implement). Omega signal levels are such that the PLL must be able to track signals at a -20 dB SNR in the 30 Hz predetection bandwidth. The loop filter (bidirectional counter) provides a narrow post-detection bandwidth of a fraction of a Hertz to enable tracking of these low-level signals. However, the increment/decrement type of phase detector and narrowband loop filter severely limit the tracking speed of the loop. Increased loop filtering gives SNR improvement but decreases the maximum tracking speed of the loop, and vice-versa.

The PLL must be capable of tracking the motion of a general aviation aircraft (nominally 200 knots velocity). The frequency offset of the receiver's local oscillator (TCXO) appears as a velocity offset in phase, thereby adding a few more tens of knots to be tracked by the PLL. This speed requirement coupled with the narrowband filtering restraint dictates the need for a higher order tracking loop (i.e. one that can track a constant velocity offset without sacrificing the needed integration). Therefore, the second-order memory-aided PLL (SMAPLL2) structure illustrated in Figure 11 has been implemented.

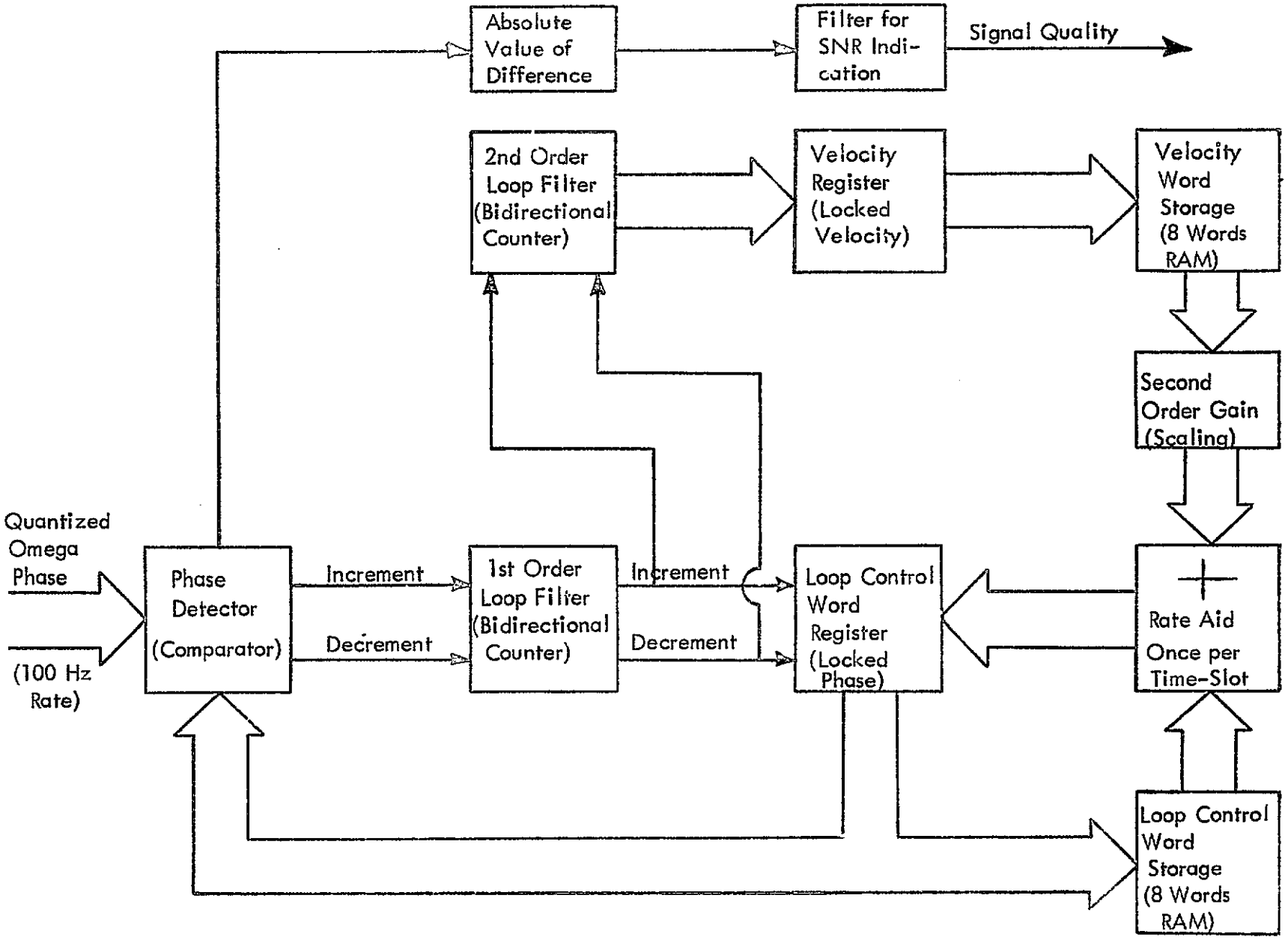


Figure 11. Second-Order Software Memory-Aided Phase Locked Loop (SMAPLL2) Block Diagram.

The operation of this PLL is similar to that of the first order loop, but a second-order loop counter and a velocity register have been added. [15] When the first-order loop counter increments or decrements the LCW, it also increments or decrements the second-order loop counter. At the end of the time-slot the contents of the second-order loop counter are added to the velocity register. The velocity register contents are then scaled by a second-order gain factor and used to rate-aid the LCW before its use in the next time-slot measurement 10 seconds later. It should be noted that the SMAPLL2 exhibits first-order loop characteristics during each time-slot, with second-order characteristics (rate-aiding) only between time-slots.

Additionally, a SNR measurement is derived from the phase detector output. As shown in Figure 11, the phase detector compares the incoming phase measurement with the LCW. This is accomplished by differencing the two numbers. This difference is a measure of how "tightly" the loop is locked, which is in turn a function of the received SNR. The absolute (unsigned) values of these differences are accumulated over all measurements in a time-slot. For a high SNR case the accumulation will be small (loop tightly locked), while for a low SNR case the sum will be large.

The SMAPLL2 parameters of first order bandwidth and second order gain have been optimized via FORTRAN simulation as documented in Appendix E. Worst-case conditions of 200 knot aircraft velocity and 100 knot TCXO clock drift in a radial direction to/from Omega stations, as well as a SNR range of -20 dB to +20 dB in the 30 Hz predetection bandwidth have been taken into account in the optimization procedure. A first order filter of 16 counts (about

.05 Hz bandwidth) combined with a second order gain of .25 have been chosen to yield the best combination of transient and steady-state responses.

Figure 12 shows SMAPLL2 transient (lock-up) response to 180 degree step and 300 knot ramp inputs. The input SNR for this Figure is 0 dB, and it can be seen that the loop requires about two and one half minutes to lock to this low-level signal. The loop requires longer times to lock to lower SNR signals and shorter times for high SNR inputs (e.g. about 15 minutes for -15 dB and 2 minutes for + dB SNR).

Figure 13 shows the steady-state response of the SMAPLL2 to the desired range of SNR's of step inputs. Figure 14 gives the same data for the 300 knot ramp inputs. The vertical axis is the standard deviation of the SMAPLL2 output estimates of phase in units of cec. Both of these Figures show that the loop's steady-state error will be less than plus or minus 3 cec 62.5% of the time whether airborne at 300 knots or on the ground motionless.

A further demonstration of the improvement in signal variance provided by the SMAPLL2 is provided by plots of the histograms for signal plus noise input phase and the loop output estimates of phase. Figure 15 gives the input histogram for a -10 dB SNR case and Figure 16 gives the loop's output histogram. The improvement made in signal variance by the SMAPLL2 is obvious from this Figure, and is directly proportional to the SNR improvement through the loop.

All SMAPLL2 parameters have been optimized for tracking in the noisy -20 dB SNR case and with the 300-knot velocity constraint imposed. Performance is considerably enhanced by higher SNR's and slower velocities. The SMAPLL2 routine requires about 150 bytes of microcomputer memory and takes about 200

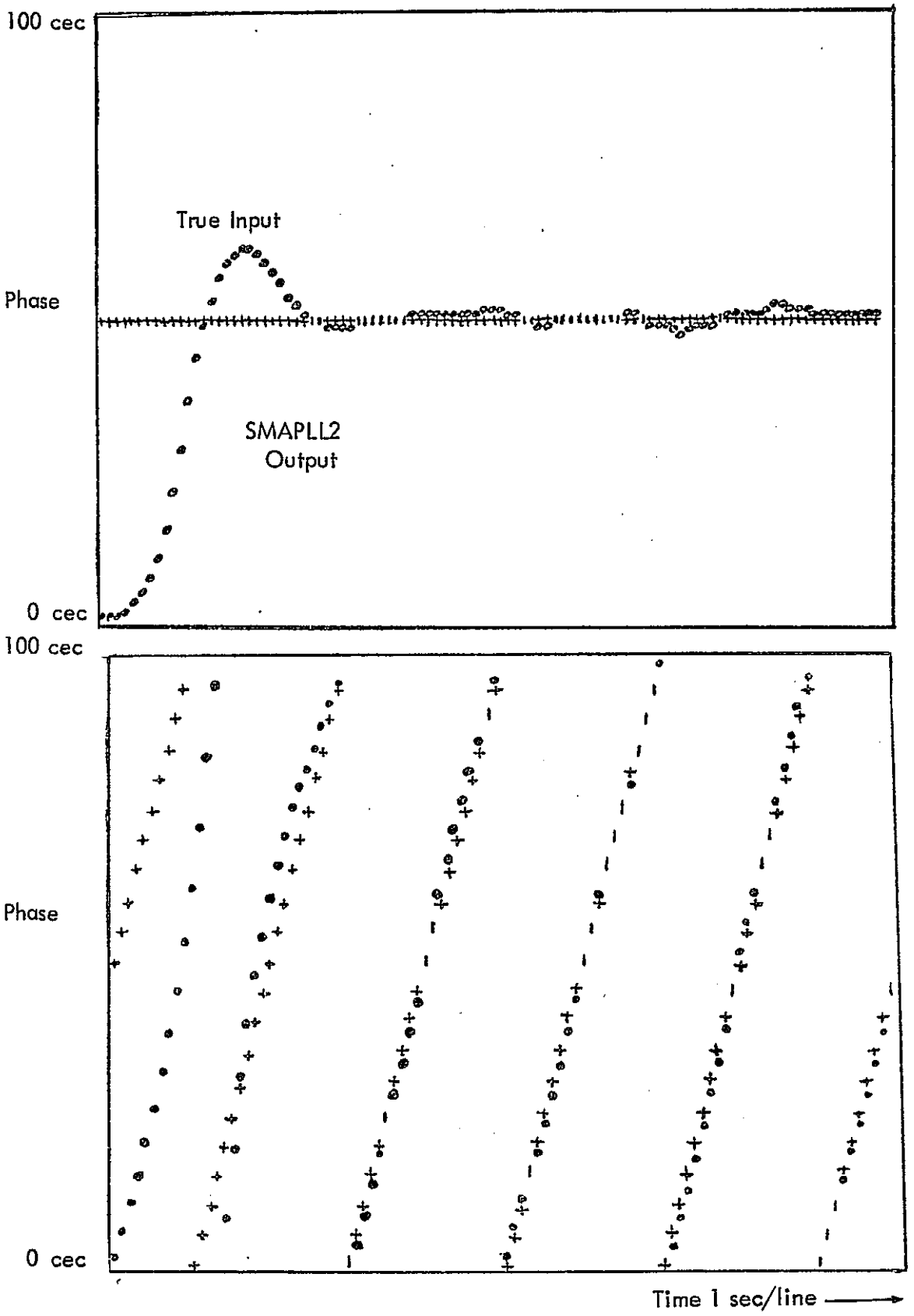


Figure 12. SMAPLL2 Transient (Lock-Up Mode) Response to 180° Step (Top) and 300-Knot Ramp Inputs (Bottom) of 0 dB SNR.

Figure 13. Steady-State Response of SMAPLL2 to Zero Velocity Inputs: Standard deviation of loop outputs in cec vs. range of inputs SNR's in dB.

ORIGINAL PAGE IS
OF POOR QUALITY

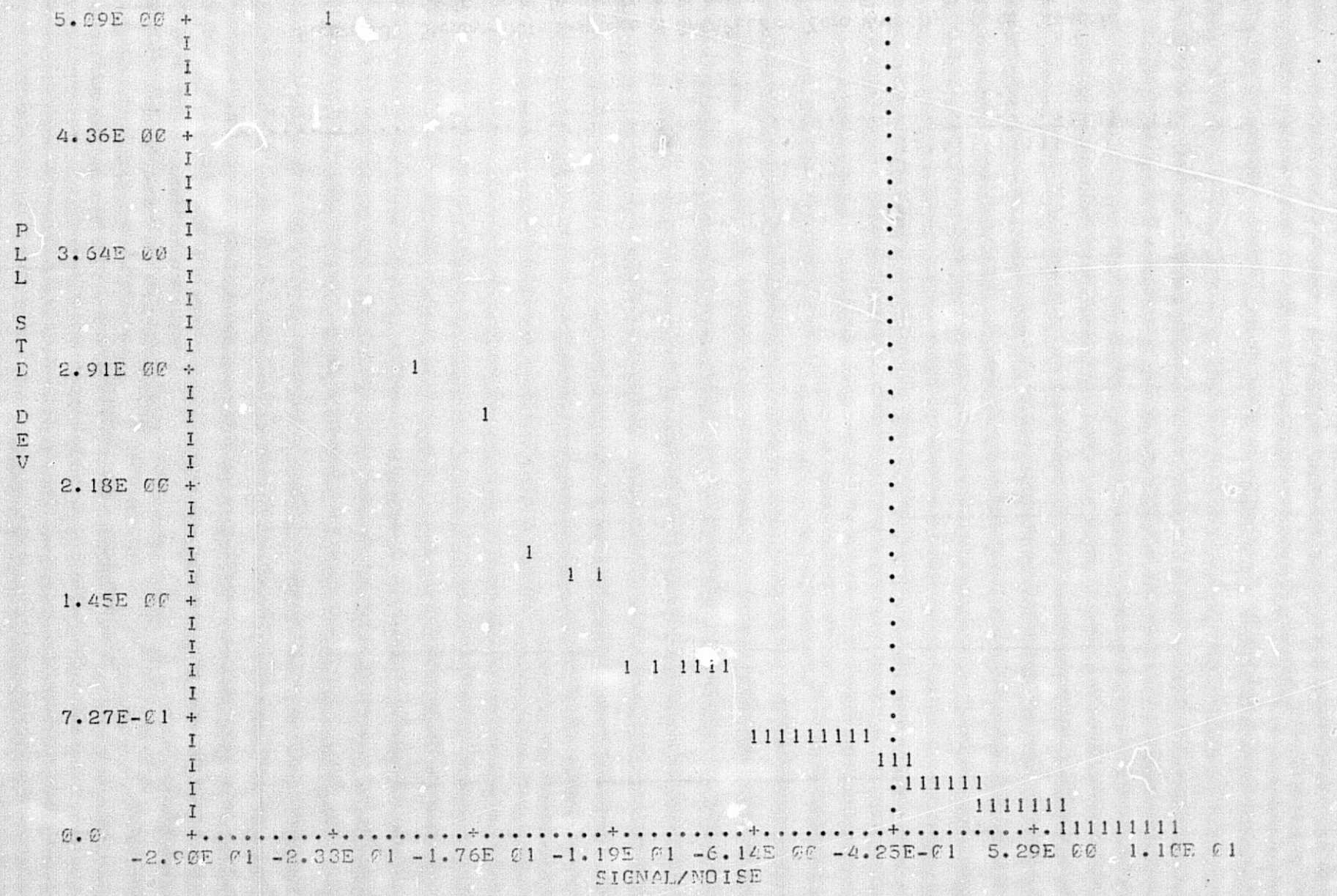
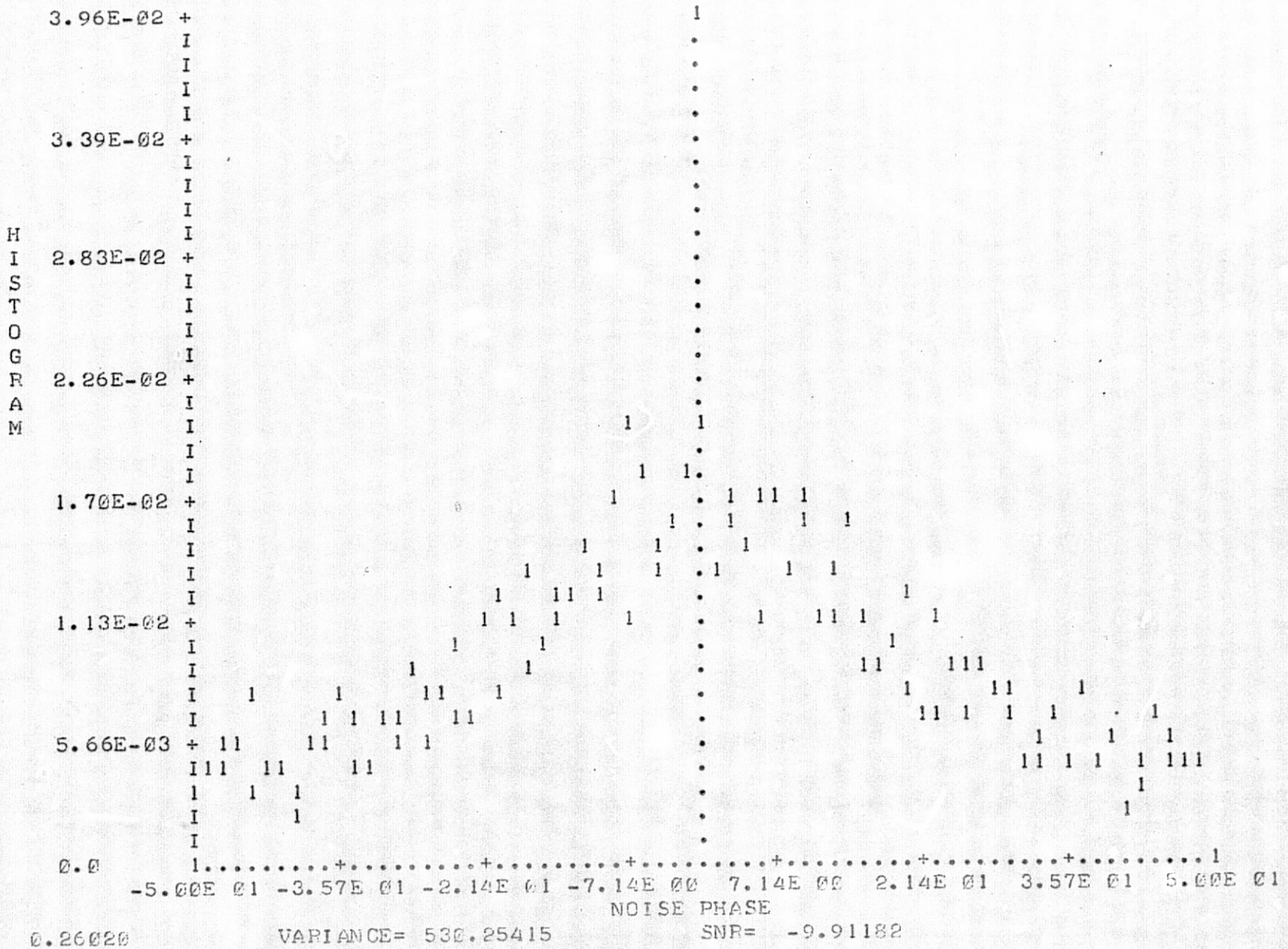


Figure 14. Steady-State Response of SMAPLL2 to 300-Knot Ramp Inputs: Standard deviation of loop outputs in cec vs. range of input SNR's.

ORIGINAL PAGE IS
OF POOR QUALITY



MEAN= 0.26020 VARIANCE= 530.25415 SNR= -9.91182

Figure 15. Normalized Histogram of -10 dB Input Signal-Plus-Noise.

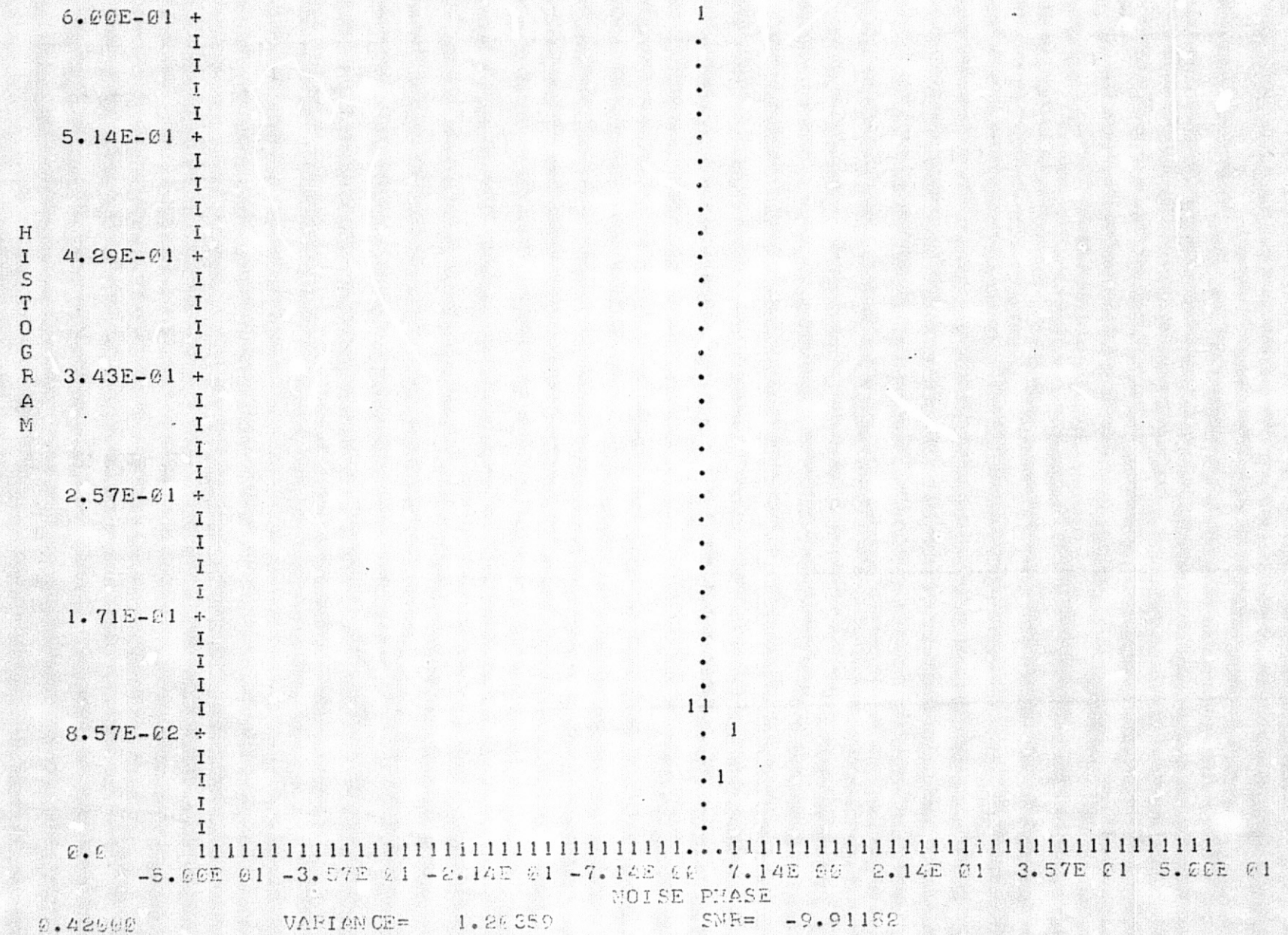


Figure 16. Normalized Histogram of SMAPLL2 Output Estimates of Phase for -10 dB Input SNR.

microseconds to execute. Appendix E gives the details of the simulation and additional plots describing the loop's performance. Appendix F gives the microcomputer program listing for the SMAPLL2.

In addition to the interrupt service routine, auto-sync subroutine, and SMAPLL2 subroutine already described (and the data output subroutine to be described in the next chapter), the OSP employs a "main", or idling, routine. At system turn-on time this routine initializes all necessary memory locations to their proper contents; it then simply idles (wasting time) until it is interrupted by the microcomputer interface module with new Omega data. This main routine would take the form of an airborne navigation program in a full-blown ONS. In its current form it requires 100 bytes of memory to initialize OSP parameters and takes about 100 microseconds to execute.

By accumulating the CPU time and program memory requirements for all the OSP algorithms presented, it is possible to estimate the CPU time and program storage currently unused (that which is available for future navigation processing algorithms). After system initialization and automatic synchronization during the first 100 seconds after turn-on, the SMAPLL2 requires about 200 microseconds after each Omega interrupt every 10 milliseconds. This leaves about 98% of the available CPU time currently unused (i.e. available for navigation processing). All OSP algorithms, including the Data Output Routine described in the next chapter, require a total of about 1000 bytes (1K) of program memory. Therefore, about 3K of the available storage is currently unused and is available for the navigation algorithms.

III. OSP OUTPUTS AND USES

A. Output Data. The Omega sensor processor receives the Omega signal off-the-air, performs automatic synchronization to the Omega transmission format, and tracks the phase of each receivable station. The data which the OSP makes available is the relative phase of each transmitting station and an indication of the relative signal strength of each station's signal with respect to the atmospheric noise level. This is the data needed by a navigation processor to derive the receiver's position.

The microcomputer-based OSP makes maximum use of the inherent design flexibility afforded by the software by utilizing a data output subroutine. The output routine is flow-charted in Figure 17, and the actual program is listed in Appendix F. The routine is built around three user-inserted code words, as illustrated in Figure 18. (Appendix D explains the insertion of data into the microcomputer). One bit of the "option" code word selects a single output or two outputs to be time multiplexed to the output latch (this latch is described in the hardware section). Another bit selects output of the phase or signal quality of the channels chosen by the second and third code words. The last six bits of the option code word dictate the multiplex time in tens of seconds to be spent on the output of each channel's data (if the multiplex option was selected by bit one). The second and third code words select the stations for which data is to be displayed. Each code word is treated as two four-bit station identifiers, and each station identifier is a number between zero and eight (corresponding to zero and the Omega stations A through H).

ORIGINAL PAGE IS OF POOR QUALITY

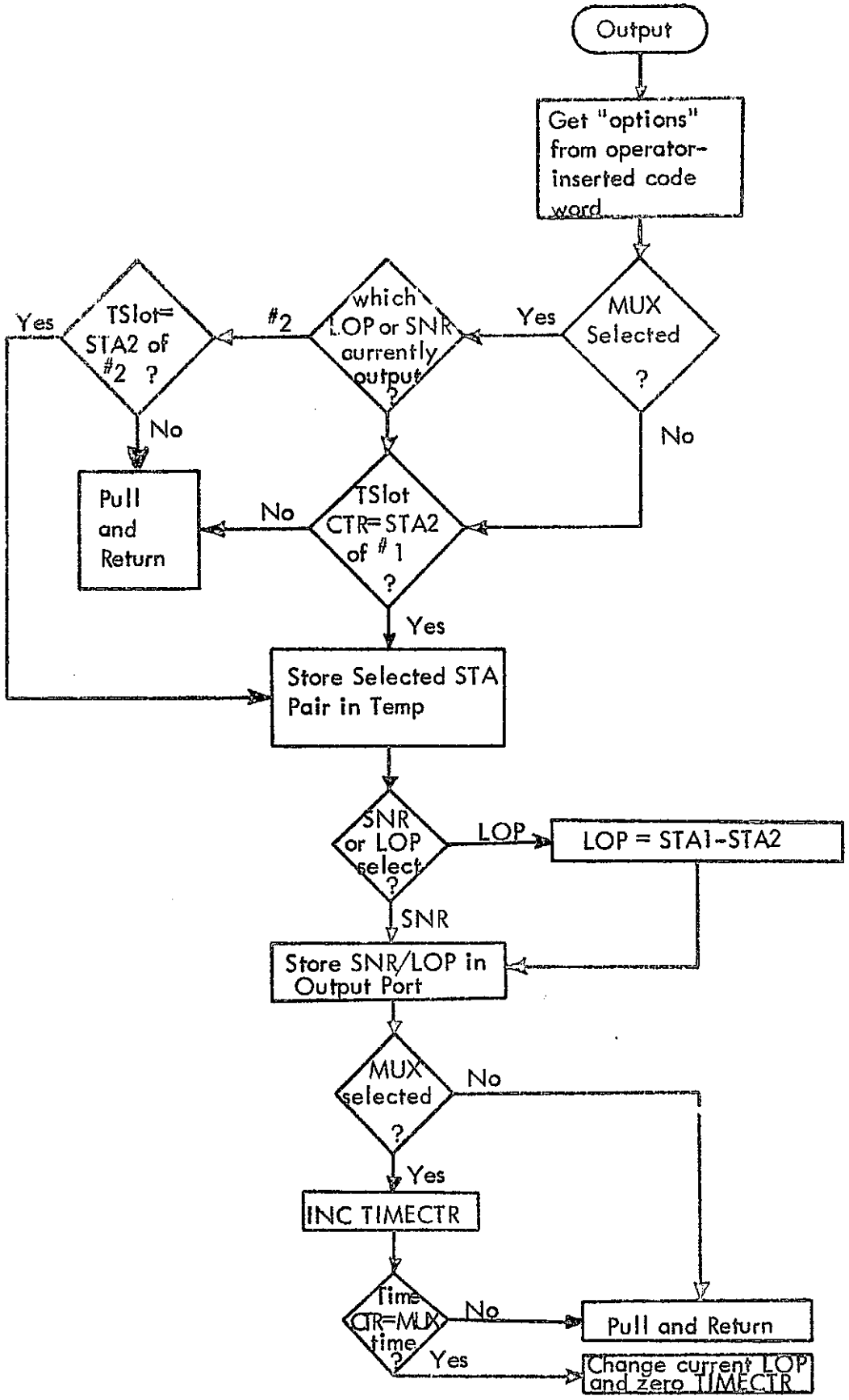


Figure 17. Data Output Routine Flow Chart.

Option Code Word	Address 0000	D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
		MUX	S/N or LOP	MUX Time					

MUX (Bit D_7) is set = 1 for multiplexing two outputs. (D_7 = 0 will result in LOP(1) or S/N(1) output only.)

S/N or LOP (Bit D_6) is set = 1 for LOP output (D_6 = 0 will result in S/N output).

MUX Time (Bits D_5 - D_0) is set to a binary number corresponding to the number of 10 second intervals to be spent outputting each LOP or S/N before switching to other LOP or S/N.

LOP and S/N Code Words	Addresses 0001 and 0002	D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
		Station 1				Station 2			

Station 1 (Bits D_7 - D_4) is a binary number between 0 and 8. The number 0 selects clock drift only; while 1-8 select Omega stations A-H.

LOP is generated by taking phase of station 1 - station 2.

If S/N is selected by making D_6 bit of option code word = 0, then the S/N of station 2 is output.

Figure 18. User-Inserted Code Words for Output Routine.

If the phase option has been selected, the numbers in the first channel code word represent the phase of station #1 minus the phase of station #2 to be output. If the number inserted for station #1 is zero, and the number of station #2 is 1-8, the phase of the station selected in station #2 with respect to the OSP TCXO will be output. This measurement of "single station phase" is exactly that which has been tracked by the SMAPLL2 tracking filter described in the software section. An example of the single station phase output appears in Figures 19 and 20. Figure 19 shows the phase of a strong SNR station (as received in Ohio) versus the OSP clock. The Omega signals are phase-locked to a stable atomic standard at the transmitter, so that the constant velocity offset in phase is caused by an offset in frequency of the OSP clock. Figure 20 shows the single station phase of a station whose received SNR is lower than in the previous graph. This signal also exhibits the ramping phase measurements, but the track is less constant or noiser than the strong station's plot.

If the phase option has been selected and neither of the numbers in the first channel word are zero, the phase difference of station #1 minus station #2 will be output. The phase difference of two Omega signals gives a line-of-position (LOP) as explained in Appendix A. Figures 21 and 22 show the LOP's plotted for various station pairs as received in Ohio. The phase ramping noted in the single station phase plots is eliminated in the LOP's by virtue of the fact that subtracting two phases measured against the same clock reference effectively cancels the clock from the measurement. Figure 21a shows the strong station pair C-D, and also illustrates the predictable diurnal phase shifting effects caused by sunrise

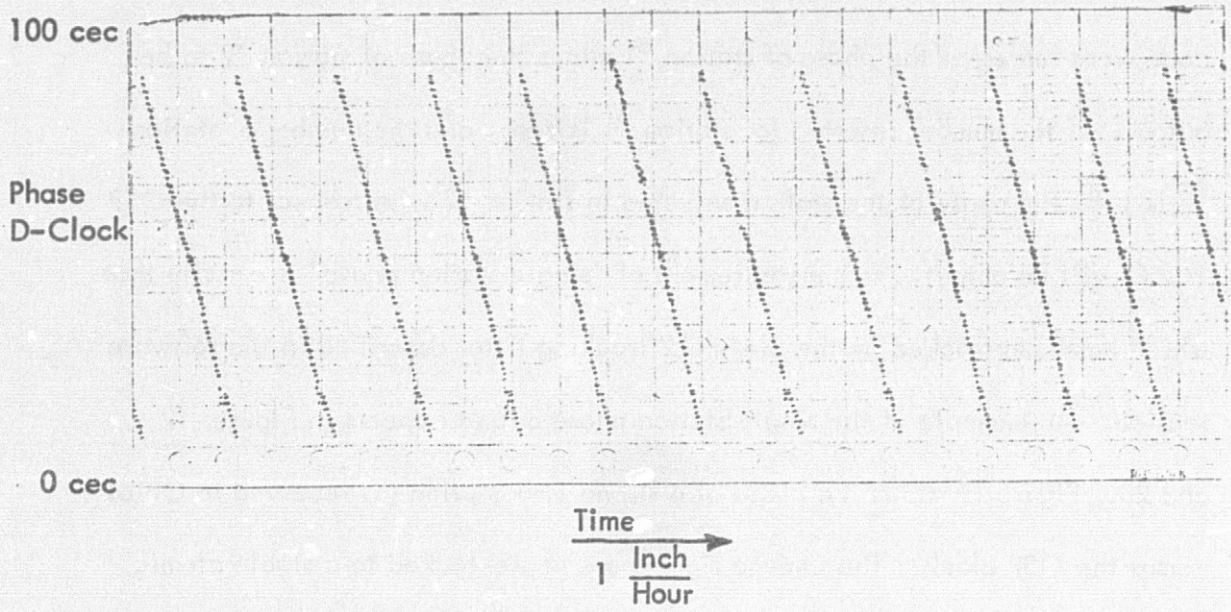


Figure 19. Single Station Phase of North Dakota Minus Clock.

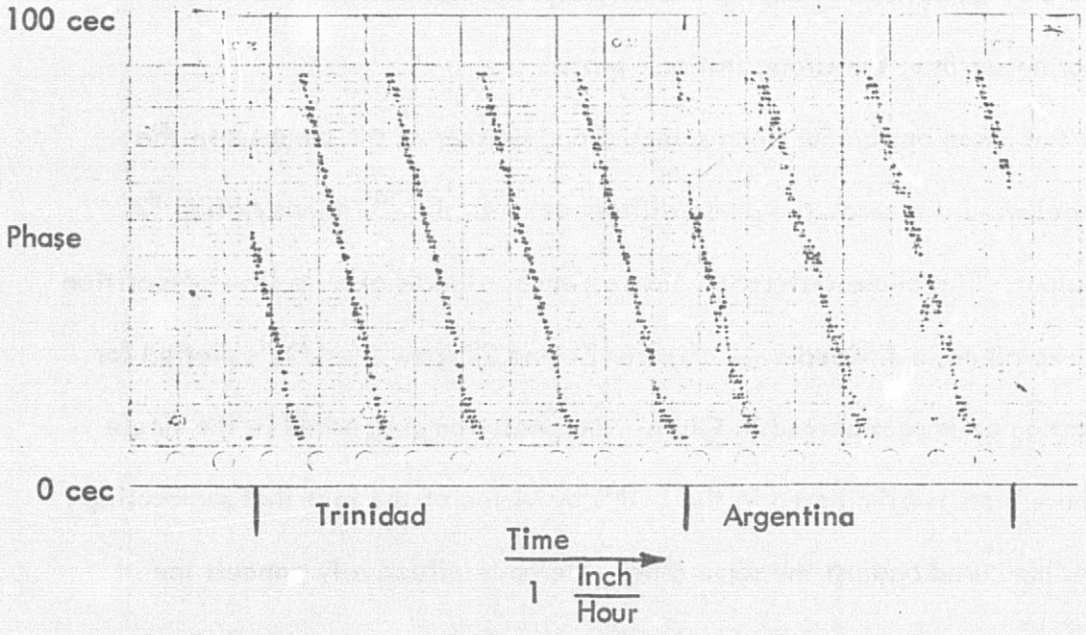


Figure 20. Single Station Phase of Trinidad Minus Clock and Argentina Minus Clock.

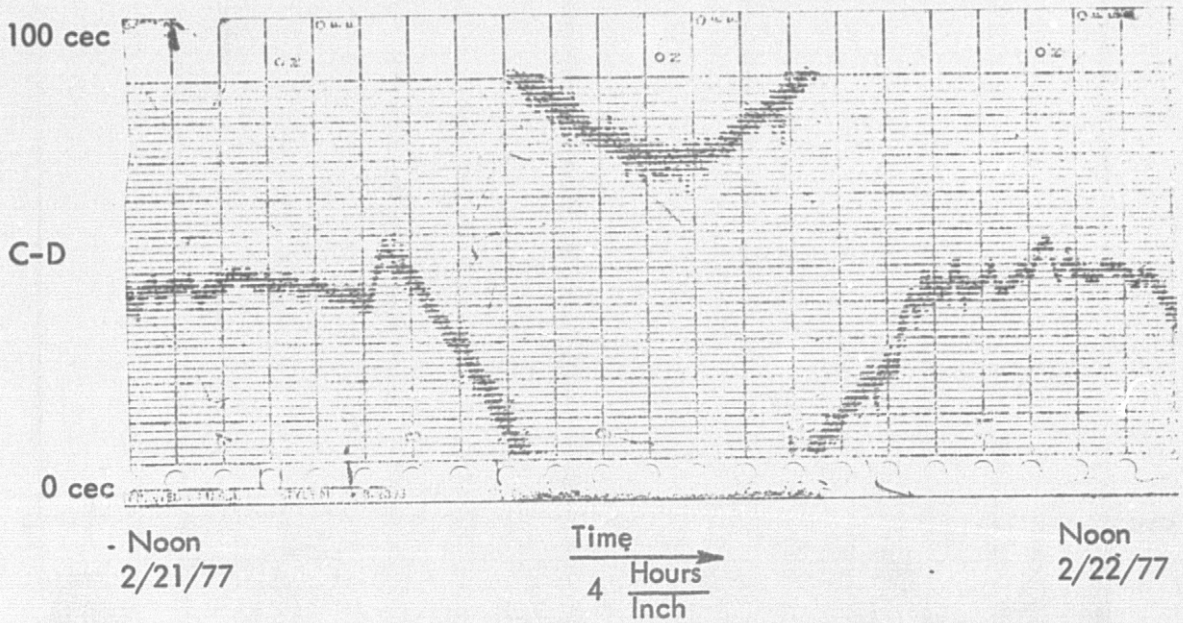


Figure 21a. 24 Hours Diurnal Data: Hawaii - North Dakota LOP.

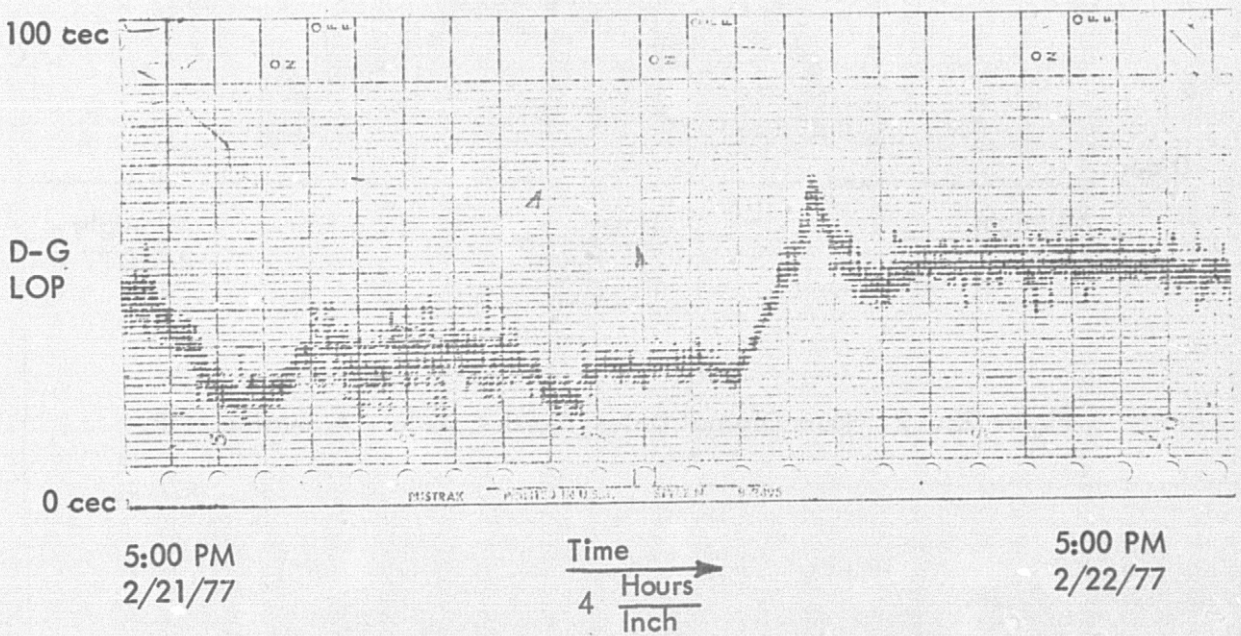


Figure 21b. 24 Hours Diurnal Data: North Dakota - Trinidad LOP.

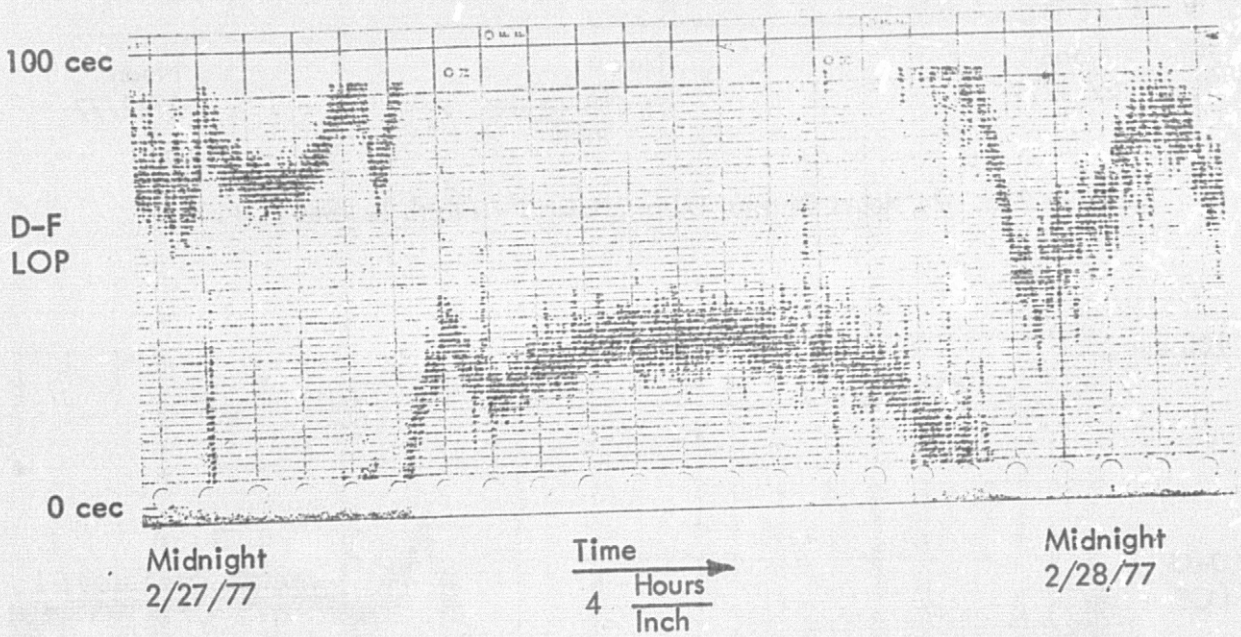


Figure 22. 24 Hours Diurnal Data: North Dakota - Argentina LOP.

and sunset. Figure 21b shows the D-G LOP which is a noisier pair, but one which is still highly usable for navigation. Figure 22 shows the D-F LOP which is very noisy, and is most useful for navigation during the atmospherically "quieter" night-time hours (when the propagation path is in darkness).

For means of comparing the software-based OSP with an all hardware OSP, Figure 23 is given which shows LOP outputs for both designs. This chart illustrates that both OSP designs track the Omega signal when in a ground-based environment. However, the second-order tracking loop employed in the software-based OSP enables the tracking of high speed aircraft with more SNR improvement than is possible with the first-order hardware version. This is verified by the computer simulation described in Appendix E.

If signal quality is selected in the option word instead of phase, the output will be the signal quality of station #2 of the first channel word. Figure 24 shows plots of the signal quality for several stations. Note that this gives only a relative SNR indication in its present form; although this measurement could be calibrated to give true SNR.

The multiplex option represents a means for time-sharing a single chart recorder or other output device. If this option is selected in the option code word, the output routine will multiplex phase or signal quality (whichever is selected) of the first and second channel code words to the output latch. The multiplex time, or time spent outputting the data for each channel word, is determined from the last six bits in the option word. This six-bit binary number represents the number of ten second intervals to be spent on each channel's data before switching to output the data indicated for the other channel word. Figure 25 gives an example

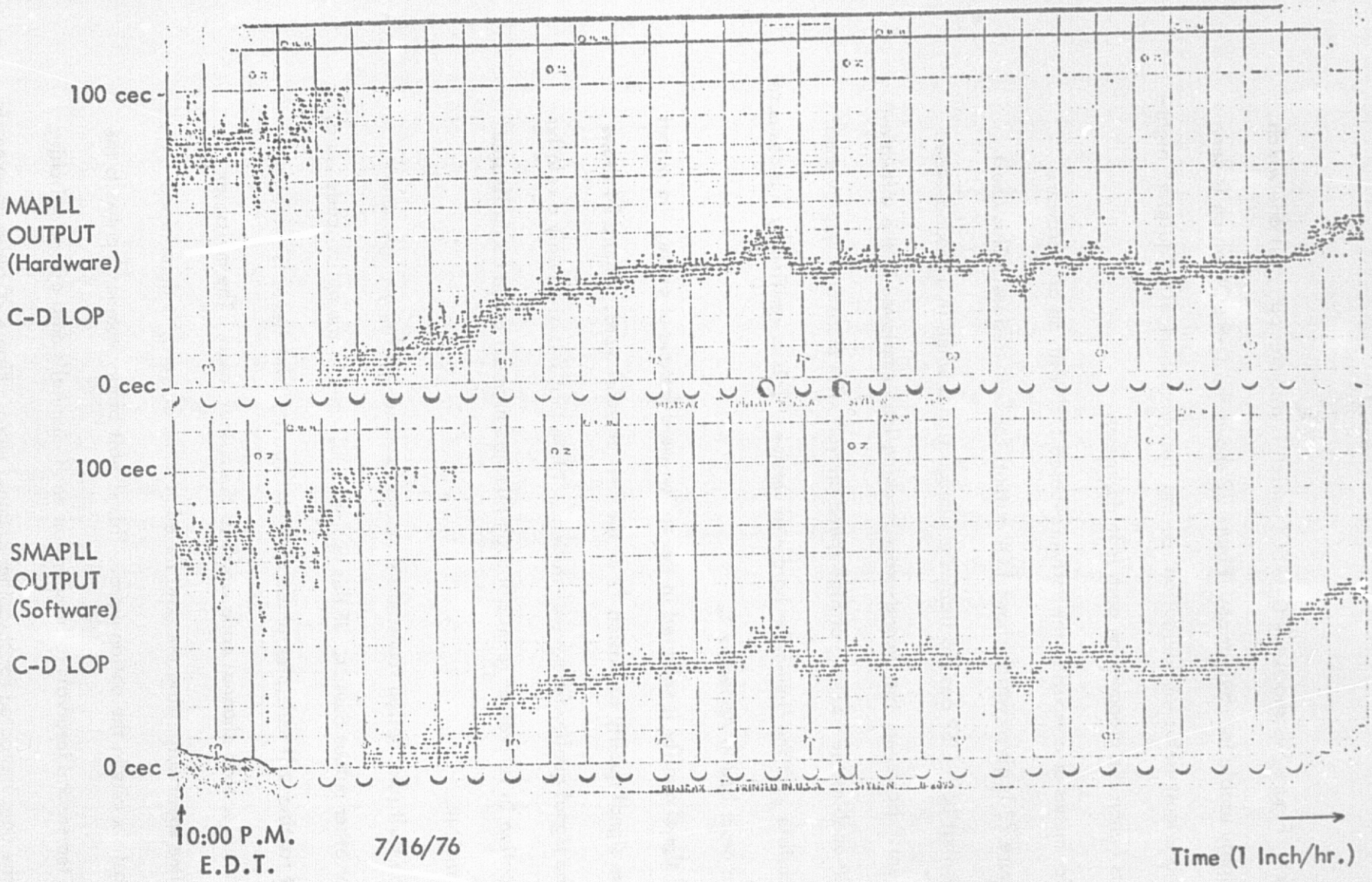


Figure 23. Omega LOP Output Traces -- Software Loop Output on Bottom.

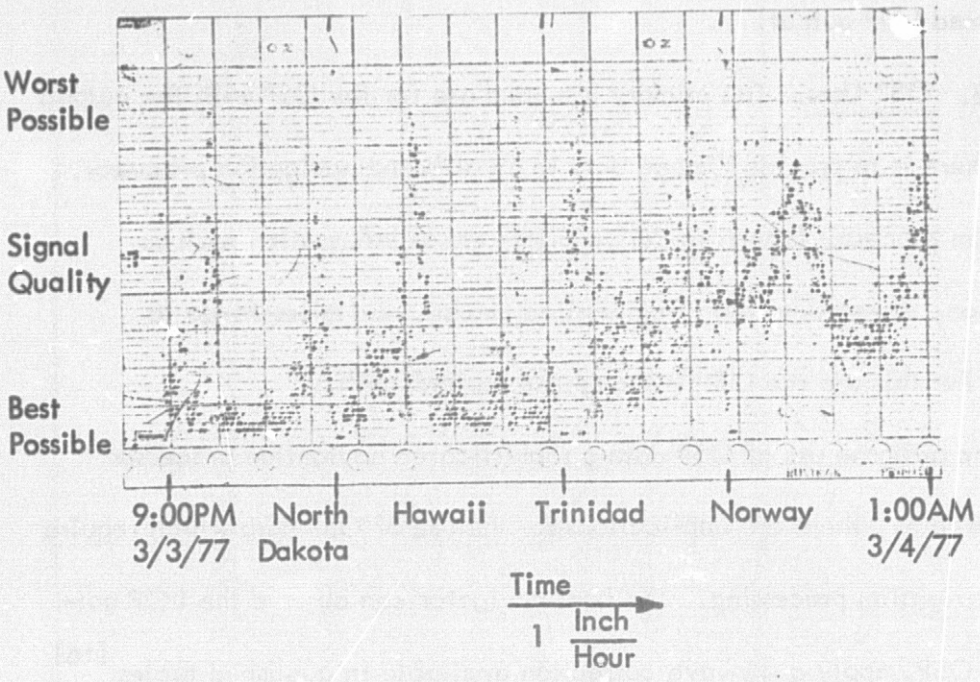


Figure 24. An Example of Signal Quality Data.

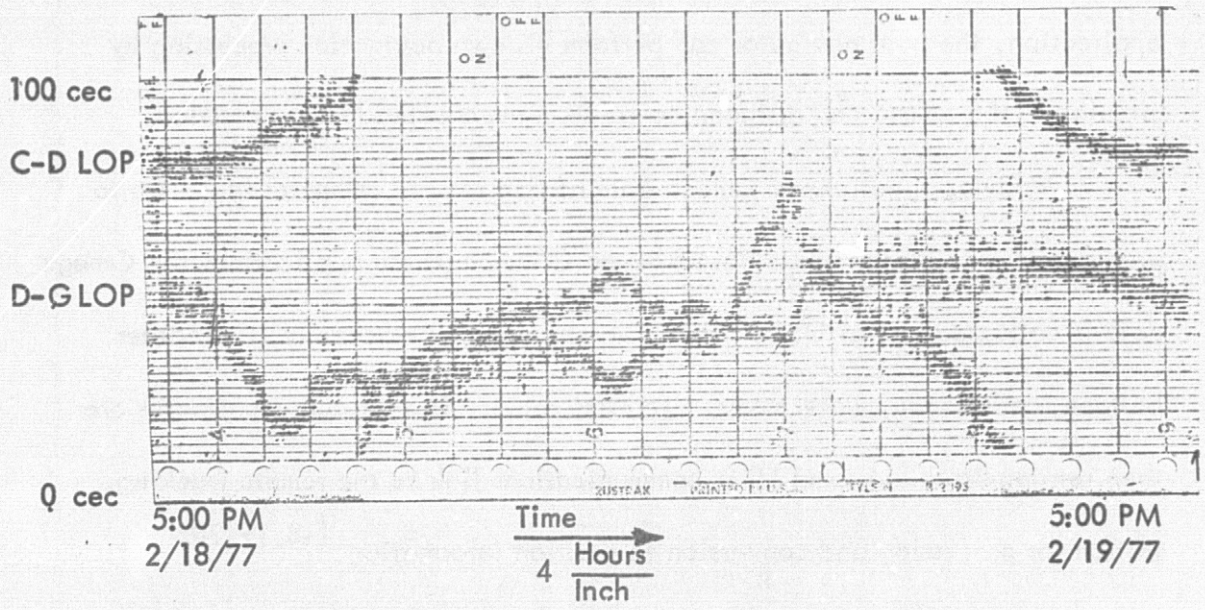


Figure 25. Multiplexed LOP Outputs (90 Seconds MUX Time).

of a multiplexed LOP output.

B. OSP Uses. The primary intended use for the OSP with the outputs as presented here is to provide Omega data to an airborne navigation processor. This navigation processor is then to derive pilot-usable information such as current position, course deviation from desired course, and miles-to-go to destination. For this use the OSP is thought to perform well.

For the airborne use of OSP data a sophisticated navigation processor is needed; however, there are applications for the use of OSP data which require little or no navigation processing. The boat navigator can observe the LOP outputs from the OSP, apply a skywave correction available in published tables [16], and find his current position on published Omega charts. [17] In this slow-speed application, the boat navigator can perform his own navigation processing by calculating the course and distance-to-go from his charted Omega position.

Many applications of Omega data utilize remote computer facilities to process the raw Omega phase data from an OSP. In these applications the Omega data is collected via an OSP on board an object to be tracked (e.g. weather balloon, radiosonde, water bouy, life raft, etc.). The Omega coordinates are then retransmitted via some other communications link to the remote tracking station for processing and conversion to position information. [18,19,20]

The U.S. Coast Guard is the U.S. government agency responsible for maintaining the Omega network in navigationally-precise, working order. [1] This entails providing a network of Omega monitor stations to receive and record the Omega signals at several geographic locations around the world for transmitter fault detection purposes and signal propagation studies. The OSP presented herein

is ideal for monitoring purposes such as this. The data can be stored on a digital or analog output device for later collection, or it can be transmitted to the monitoring headquarters via phone lines in near real time.

Finally, the single station phase measurements such as illustrated in Figure 19 can be used to calibrate laboratory oscillators to near atomic clock precision. Since the Omega signals are locked to atomic standards at the transmitters, the received signal versus local clock measurement shows the offset of the local clock. By adjusting the frequency of the local clock until the phase ramping gives way to a straight line (constant phase measurement) the local oscillator can be calibrated.

IV. SUMMARY AND CONCLUSIONS

A microcomputer-based low-cost Omega sensor processor for general aviation use has been designed, implemented with minimum hardware and innovative software and tested. The sensor processor hardware configuration has been presented and the software has been described.

A FORTRAN simulation analysis of the functions of automatic synchronization and PLL phase tracking has been performed to optimize routines to be performed by the OSP microcomputer. A data base of real Omega data was collected using the system's receiver and interface modules and was used as the simulation input to obtain realistic results.

The microcomputer routines developed have been shown to output data comparable to an all-hardware OSP and, in addition, the software-based OSP offers data options and capabilities not available with the all-hardware model. The OSP provides outputs suitable for use by a navigation processor, and the

system presented includes nearly all the hardware needed for an airborne Omega navigation system.

The feasibility of implementing a minimum hardware software-based OSP of higher performance and lower cost than is currently available has been demonstrated. Further research is encouraged to develop navigation processing microcomputer software to complete the Omega navigation system for general aviation use.

V. RECOMMENDATIONS FOR FUTURE WORK

The OSP presented herein represents nearly all the hardware needed for a complete Omega navigation system. The software OSP routines provide Omega data usable by a navigation processor to derive pilot-usable information. Needed to complete the system for general aviation use is a microcomputerized area navigation routine. One possible navigation technique using Omega data has been functionally described by Zervos.[21] In addition, a propagation correction routine or some other means for applying skywave corrections is needed. More work also needs to be done to develop a lower-cost, effective pilot display.

A software-based navigation system is afforded inherent flexibility in that the computer memory can be reprogrammed to perform other functions. The OSP design presented can be used for some other navigation system by augmenting or replacing the Omega front-end module with a LORAN, GPS, or other front-end and modifying the software. The potential for a hybrid Omega-LORAN system seems great, considering the possible sharing of system hardware and software by the two systems.

Aside from the noted receiving system developmental work, additional research is needed to determine means for improving the redundancy and reliability offered by the Omega transmitting network. Whether this takes the form of a supplemental navigating system in the receiver-computer or an actual transmitter station augmentation, pilot confidence in an area navigation system must be justified and earned. Thereafter, a means for efficiently integrating Omega into the National Airspace System needs to be pursued.

VI. ACKNOWLEDGEMENTS

The author wishes to thank Mr. Jack Reid of the NASA/Langley Research Center for his support of this project as the Project Monitor of the NASA Tri-University Program. Gratitude is expressed to Dr. Richard McFarland, Director of Avionics at Ohio University for his support and encouragement. Dr. J. E. Essman of the Electrical Engineering faculty is also acknowledged for his constructive advice.

Special appreciation is reserved for Mr. Ralph Burhans, Avionics Research Engineer, and Dr. Robert Lilley, Assistant Director of Avionics, without whose technical expertise and enthusiastic support this thesis would not have been possible. Mr. Kent Chamberlin and Mr. Paul Blasche, Graduate Research Associates, have the author's sincere thanks for their technical consultation.

The author's Tri-University Program student colleagues at Ohio University, Princeton University, and MIT are also acknowledged as being instrumental in this project's completion. Thanks also go to Mrs. Roma Beverage and Mrs. Hope Mills for their work on the manuscript.

VII. REFERENCES

[1] Department of Transportation, U.S. Coast Guard, Omega Navigation System User Manual, CG-ONSOD, 2100 2nd St. SW, Washington, D.C., preliminary draft in January, 1976.

[2] Pierce, J.A., et.al., Omega: A World-Wide Navigation System: System Specification and Implementation, AD 630 900, National Technical Information Service, Springfield, Virginia, issued May, 1966.

[3] Morris, Peter, and Milton Cha, Omega Propagation Corrections: Background and Computational Algorithm, Report No. ONSOD-01-74, Department of Transportation, U.S. Coast Guard, Washington, D.C., December, 1974.

[4] Radio Navigation Group, Northrop Corporation, Omega Navigation Set AN/ARN-99 (XN-2), Phase II, Final Engineering Report, 2301 W. 120th St., Hawthorne, California, December, 1973.

[5] Ohio University Avionics Engineering Center, Prototype Omega Receivers for Use in Data Collection and Evaluation, Athens, Ohio, September, 1976.

[6] Burhans, R.W., Low-Cost, High-Performance, VLF Receiver Front-End, Technical Memorandum (NASA) 18, Avionics Engineering Center, Ohio University, Athens, Ohio, September, 1975.

[7] Wright, Lee, Improvements for Omega RF Preamplifiers, Technical Memorandum (NASA) 24, Avionics Engineering Center, Ohio University, Athens, Ohio, April, 1976.

[8] Lilley, R. W., and R. J. Salter, Simulation Analysis of a Microcomputer-Based, Low-Cost Omega Navigation System, Proceedings of the Bicentennial National Aerospace Symposium of the Institute of Navigation, Warminster, Pennsylvania, April, 1976.

[9] Lilley, R. W., and R. J. Salter, A Microcomputer-Based Low-Cost Omega Navigation System, Proceedings of the First Annual Meeting of the International Omega Association, Arlington, Virginia, July, 1976.

[10] Salter, R. J., "Navigation with Mini-O, Part 3, Software", Byte Magazine, Peterborough, New Hampshire, Vol. 2, No. 4, April 1977.

[11] MOS Technology, Inc., MCS6500 Microcomputer Family Hardware Manual, 950 Rittenhouse Road, Norristown, Pennsylvania, August, 1975.

[12] MOS Technology, Inc., MCS6500 Microcomputer Family Programming Manual, 950 Rittenhouse Road, Norristown, Pennsylvania, August, 1975.

[13] Chamberlin, K. A., Digital Correlation Detector for Low-Cost Omega Navigation, Technical Memorandum (NASA) 19, Avionics Engineering Center, Ohio University, Athens, Ohio, February, 1976.

[14] Raab, Frederick H., and Jerome R. Waechter, The Use of a Counting Phase Detector to Preprocess VLF Radionavigation Signals, to be published in IEEE Transactions on Aerospace and Electronics Systems.

[15] Chamberlin, K. A., and R. W. Lilley, The Memory-Aided Phase-Locked Loop: A Model and Proposed LSI Chip Implementation, Final Report prepared for Naval Avionics Facility, Indianapolis, Indiana, No. EER 26-1, Avionics Engineering Center, Ohio University, Athens, Ohio, March 1976.

[16] H.O. Pub.No. 224 (111-C) Series: Omega Propagation Correction Tables, U.S. Naval Oceanographic Office, Washington, D.C., 1971.

[17] Omega Navigational Charts, 7500 Series, U.S. Naval Oceanographic Office, Washington, D.C.

[18] Beukers, John M., Windfinding Using Navigational Aids, presented at The Third Symposium on Meteorological Observations and Instrumentation of The American Meteorological Society, February, 1975.

[19] Baker, Donald L., and Christopher S. Welch, A Prototype Legrangian Current Buoy Using the Carrier Plus Sideband (CSB) Retransmission of Omega Navigation Signals, Proceedings of the First Annual Meeting of the International Omega Association, Arlington, Virginia, July, 1976.

[20] Rupp, Walter E., GRAN: Experiences and Activities, Proceedings of the First Annual Meeting of The International Omega Association, Arlington, Virginia, July, 1976.

[21] Zervos, Arthouros, K., A Performance Simulation of a Potential Class of Low-Cost Omega/Air Data Navigation Systems, Masters Thesis, Princeton University, Department of Aerospace and Mechanical Sciences, September, 1975.

[22] JOLT CPU Hardware Manual and JOLT DEMON Software Manual, Microcomputer Associates, Inc., Santa Clara, California, First Edition, 1975.

[23] Lilley, Robert W., An Assembler for the MOS Technology 6502 Microprocessor as Implemented in the JOLT (TM) and KIM-1 (TM), Technical Memorandum (NASA) 44, Avionics Engineering Center, Ohio University, Athens, Ohio, November, 1976.

[24] Schwartz, Mischa, Information Transmission, Modulation and Noise, Second Edition, McGraw Hill Book Company, New York, 1970.

VIII. APPENDICES

A. The Worldwide Omega Navigation System. The worldwide Omega System is a network of very low-frequency (VLF) transmitters located at eight locations around the world, as shown in Figure A-1. Each station transmits for approximately one second out of each ten seconds on the frequencies of 10.2, 13.6, and 11.333 KHz. As shown in Figure A-2, a unique transmission pattern is formed by the scheduled length of each station's transmission.

Omega was originally conceived by Professor J. Pierce of Harvard University following his successful demonstration of the long-range phase stability of VLF signals in the 1940's. Originally developed by the U.S. Navy for use on seagoing ships, Omega has only recently been used in airborne applications. Operational responsibility for the system was transferred to the U.S. Coast Guard in 1971, a move indicative of increasing civilian use of Omega. As of this writing, seven of the eight permanent transmitters are in operation, and the Coast Guard is beginning implementation of approximately forty monitor receiving stations around the world.

Since the wavelength of a 10.2 KHz signal is about 16 nautical miles, an effective antenna for radiating this frequency must be on the order of several miles long. Two types of antenna structures are utilized for Omega transmission. As diagrammed in Figure A-3, these are the tower and valley span antennas. Signals in the VLF region propagate in the waveguide mode using the conducting earth and ionosphere as waveguide bounds. As shown in Figure A-4, a single waveguide mode dominates when several hundred miles from a transmitter, and due to this single, stable mode of propagation the VLF signals can be received in

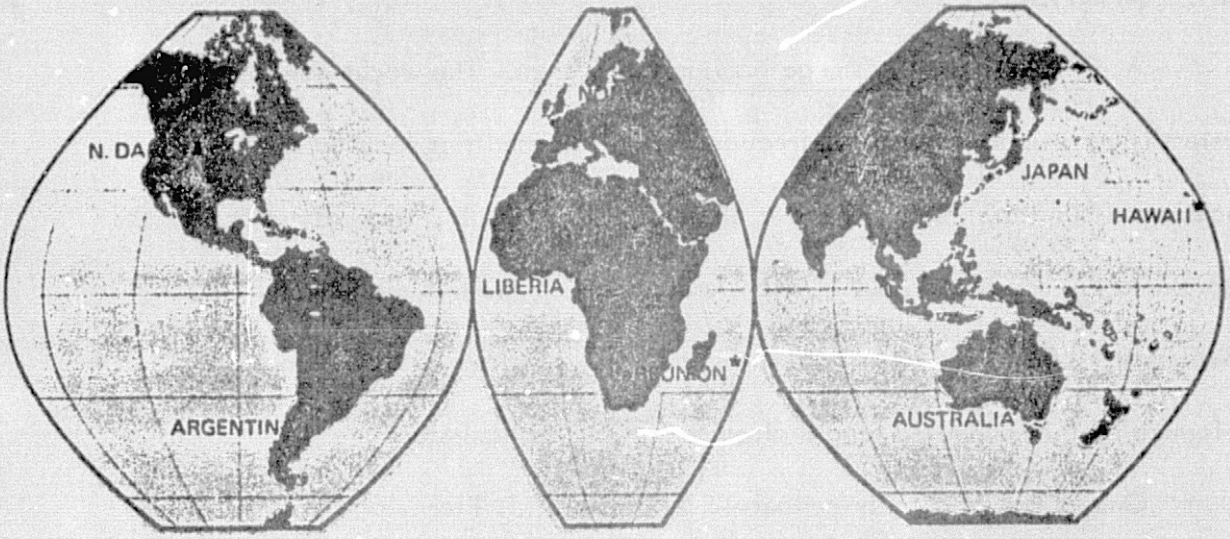


Figure A-1. Omega Station Locations.

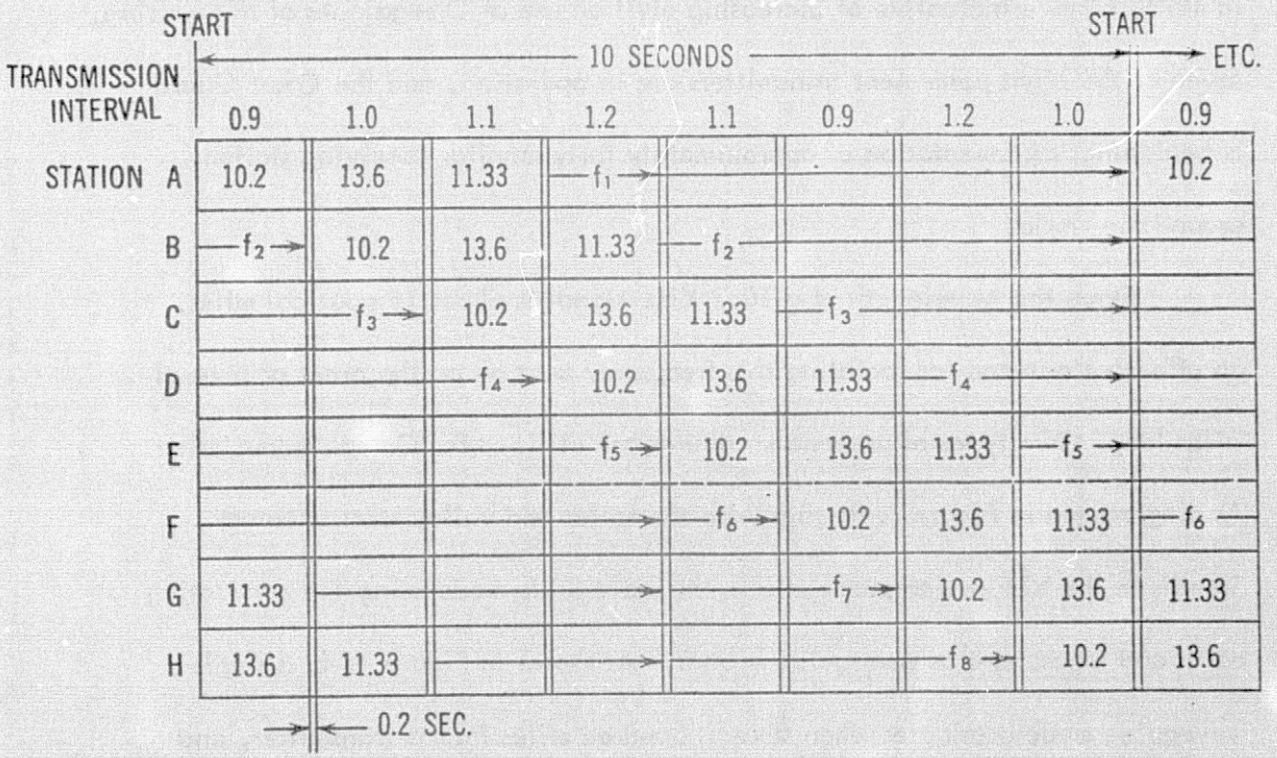


Figure A-2. Omega Signal Transmission Format.

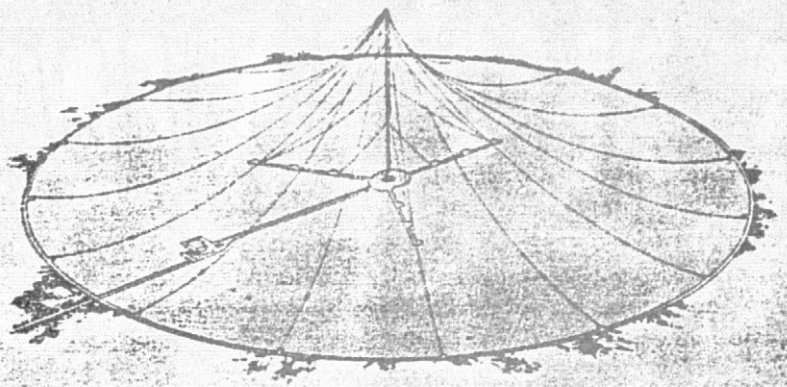


Figure 3
North Dakota Omega Transmitting Station (Typical Tower)

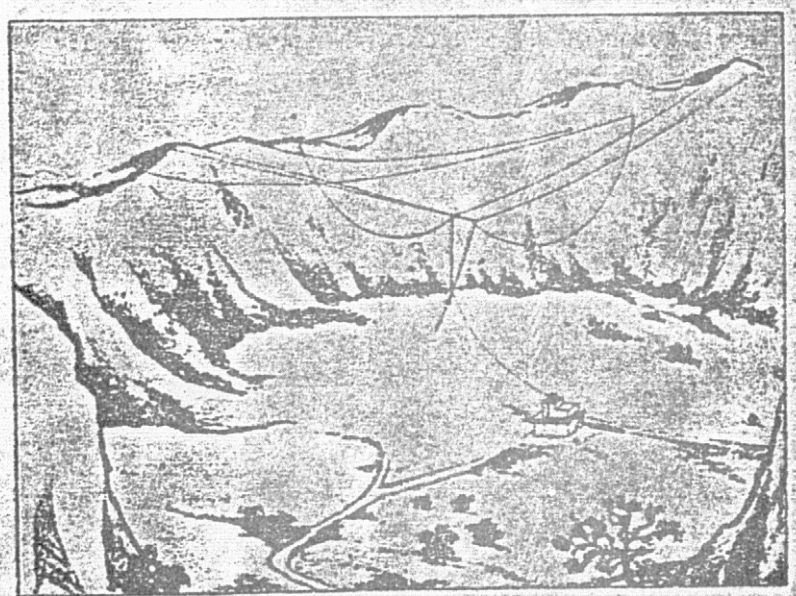


Figure 4
Haiku, Hawaii Omega Transmitting Station (Valley Span)

Figure A-3. Omega Transmitting Antenna Structures.

ORIGINAL PAGE IS
OF POOR QUALITY

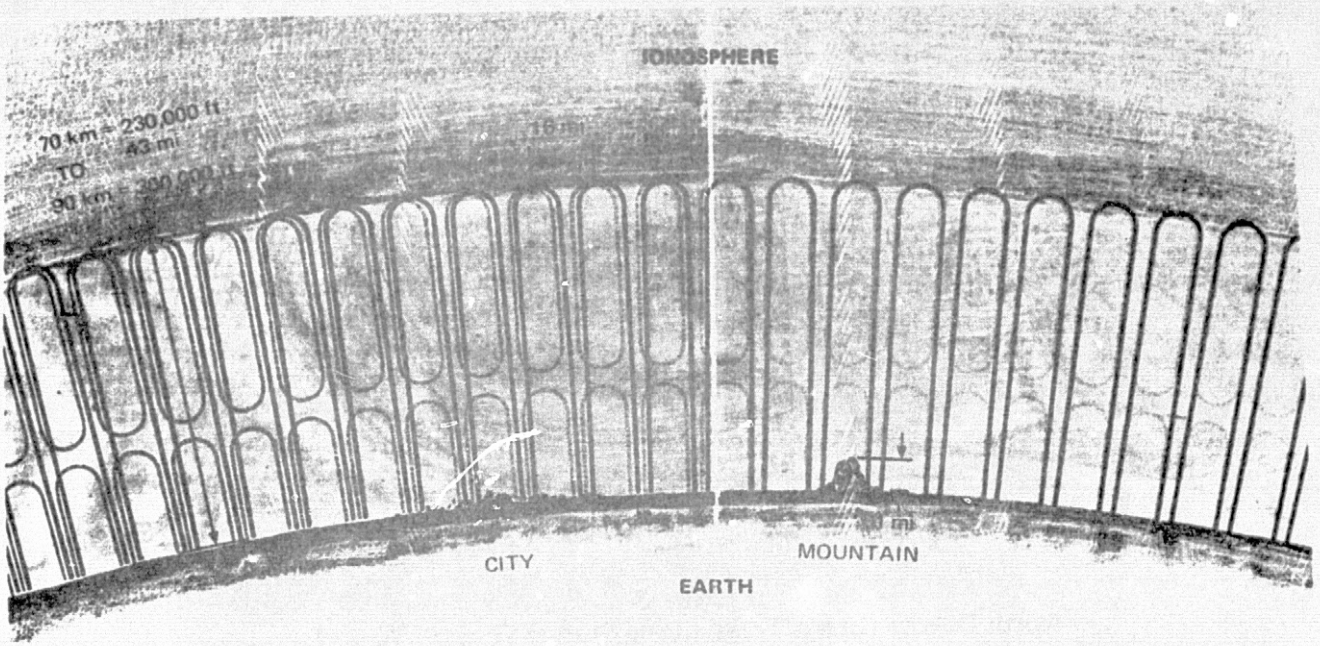


Figure A-4. Propagation Modes for VLF Signals.

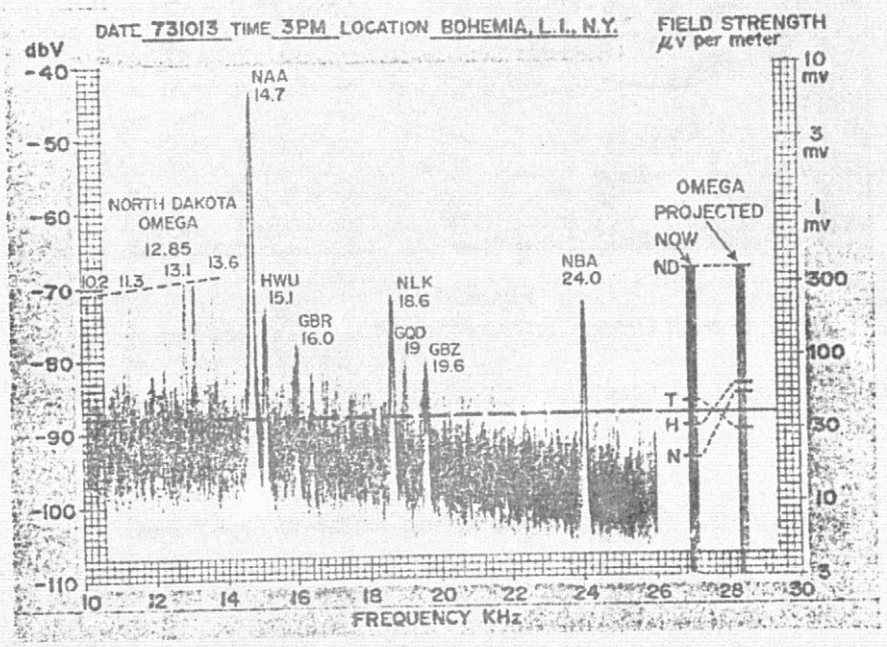


Figure A-6. VLF Signal Levels as Received in 30 Hz Bandwidth in New York.

excess of 6,000 miles and at all altitudes (even between mountains and beneath the surface of water).

The height and smoothness of the ionosphere boundary is changed by varying amounts of radiation received from the sun, and this effective change in waveguide height results in a signal phase velocity change. The phase velocity change occurs when the signal propagation path to the receiver changes from light to darkness and vice-versa, and is, therefore, referred to as "diurnal" shift. This diurnal shift is highly predictable, and the received phase can be adjusted by applying a skywave correction (SWC) factor which is unique for month, day, time-of-day, and receiver location. Another highly predictable factor is the varying conductivity of the earth waveguide boundary. Ground conductivity maps have been prepared and a propagation correction can be applied for particular receiver location. Ionospheric disturbances caused by sunspot activity and solar flares are not as highly predictable and limit the position-fix accuracy attainable with Omega.

Techniques for the reception of the Omega carrier signal differ; however, detection of the signal's phase involves comparing the received signal to a locally generated signal of the same frequency. Differing degrees of sophistication are possible in the phase detection process; more sophisticated meaning more accurate local clock signals. If a clock frequency on the order of a part in 10^{12} (atomic standard accuracy) is available, and it is possible to synchronize the phase of the local clock to the time standards employed at the Omega transmitters, the Omega system can be used in the rho-rho positioning mode by simply measuring the time-of-arrival of two or more Omega signals (propagation time being

proportional to distance traveled). If a local clock of this high accuracy is available, but it is not possible to synchronize its phase with the Omega time standards, it is still possible to navigate in the rho-rho mode. However, it is first necessary to determine the receiver's position within ±eight nautical miles (since a cycle at 10.2 KHz is 16 nautical miles long and each phase repeats every 16 miles). This is known as lane resolution, and several techniques exist for resolving lanes to within needed limits.

If a local clock of atomic standard accuracy is not available, a less stable (and less expensive) clock may be used provided short-term stability constraints are obeyed. For example, a clock stability of one part in 10^5 per day will drift no more than one centicycle (1/100 of a 360 degree cycle) in ten seconds. Therefore, if this local reference clock is, in effect, used within each ten second Omega frame separately, very little accuracy is sacrificed. Specifically, phase differencing can be performed by first detecting each station's signal with respect to the local clock and then subtracting two phase measurements. This effectively subtracts the clock from the measurement. Each phase difference is a point on a hyperbolic contour. Geometrically, lines of constant phase difference form hyperbolic contours referred to as hyperbolic lines-of-position (LOP's), as shown in Figure A-5. Navigation by reference to these LOP's is known as the hyperbolic mode, and it is again necessary to determine initial position to within one-half lane (lane resolution). It is necessary to use two LOP's (at least three stations) to determine position, and greatest accuracy is achievable when using LOP's which intersect at close to a 90 degree angle.

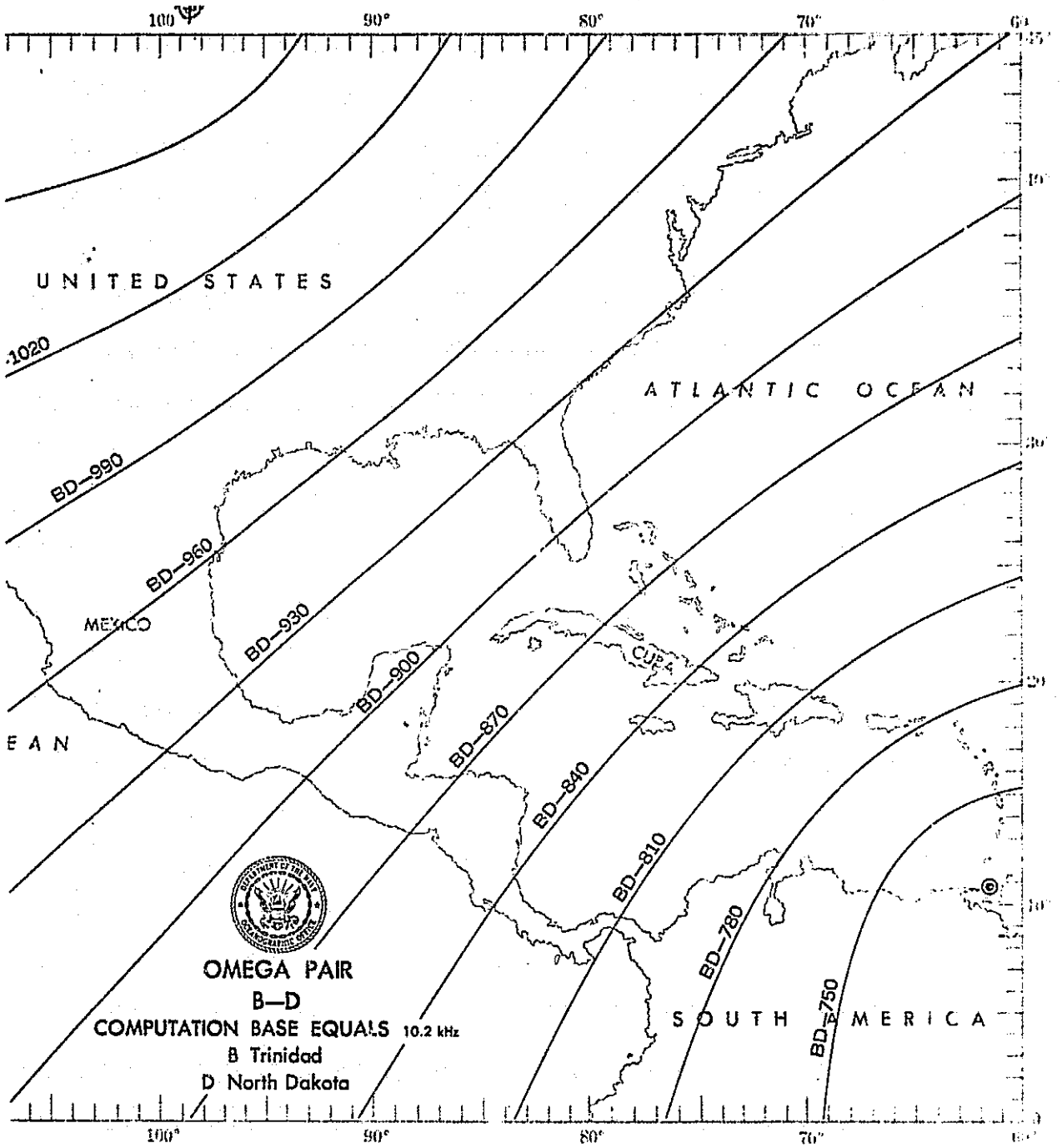


Figure A-5. Omega Lines-of-Position (LOP's) Formed by North Dakota (D) and Temporary Station Trinidad (B).

Paramount in the detection and estimation of Omega phase is the problem of noise. Amplitude noise in the VLF region is caused primarily by lightning and is made up of both gaussian distributed noise (a conglomeration of many distant lightning sources of various strengths and distances) and impulsive noise (closer lightning sources). Transmitted power of all Omega stations is ten kilowatts, and a representative sample of received signal and noise amplitudes appears in Figure A-6. Also of importance is the noise caused by precipitation upon the aircraft and antenna. These additive amplitude noises cause the detected phase to be contaminated with noise (i.e. phase jitter). Due to the several contributing noise sources and allowing for less than perfect propagation corrections, the stated Omega system position fix accuracy is + - one nautical mile (daytime) to + - two nautical miles (nighttime). This basic accuracy limitation constrains Omega use to an area (non-precision) navigation aid.

B. Omega Data Base. For purposes of obtaining realistic data for a simulation analysis of Omega sensor processor (OSP) functions, a data base of real Omega ground and flight data was collected. So that the data used as input to the simulation would be the same as that seen by the OSP microcomputer algorithms, the data was collected using the OSP hardware front-end and interface modules which had been fabricated and perfected before the OSP software was developed.

For purposes of evaluating the performance of the simulated automatic synchronization and PLL tracking processes it was necessary to collect Omega data under the following conditions: (1) ground-based receiver with noise conditions found throughout a 24-hour day; (2) ground-based receiver with and without the overwhelming strong station (North Dakota) being received; and (3) airborne receiver in a general aviation aircraft.

The Omega phase data was collected via the equipment set-up as shown in Figure B-1. The microcomputer provides an interrupt signal when new data is latched and ready to be read. For data collection the 100 Hz interrupt signal was used as a magnetic tape recorder "write" signal (one phase sample written every ten milliseconds). In addition, it was possible to collect data only during every other ten second time frame. This was to allow time to write an end-of-record on the tape to separate the ten second data frames. Note that this is wideband 10.2 KHz Omega phase data taken directly from the 30 Hz bandwidth receiver front-end and sampled by the interface module every ten milliseconds.

The ground-based data was collected at Ohio University's Clippinger Laboratories. A 6.1 meters long wire, mounted on the roof of Clippinger Laboratories was used as the receiving antenna. An active preamplifier was colocated at the antenna and feeds the signal through approximately 30.5 meters of coaxial lead-in cable to the receiver in the Omega Laboratory. The electrical environment of this receiving antenna is one of considerable interfering noise. The noise is primarily from two sources: (1) 60 Hz harmonics radiated by high-powered electrical equipment in the building; and (2) audio modulation of a nearby AM radio transmitter. Therefore, the ground-based data is of worst-case variety. The data without strong station North Dakota was collected as that station was going off-the-air for scheduled maintenance.

The airborne data was collected with the equipment set-up of Figure B-1 installed onboard the Avionics Engineering Center DC-3 flying laboratory. The data was collected outbound to Chicago O'Hare International Airport from Ohio University Airport at Albany, Ohio (and also inbound on the return flight). The

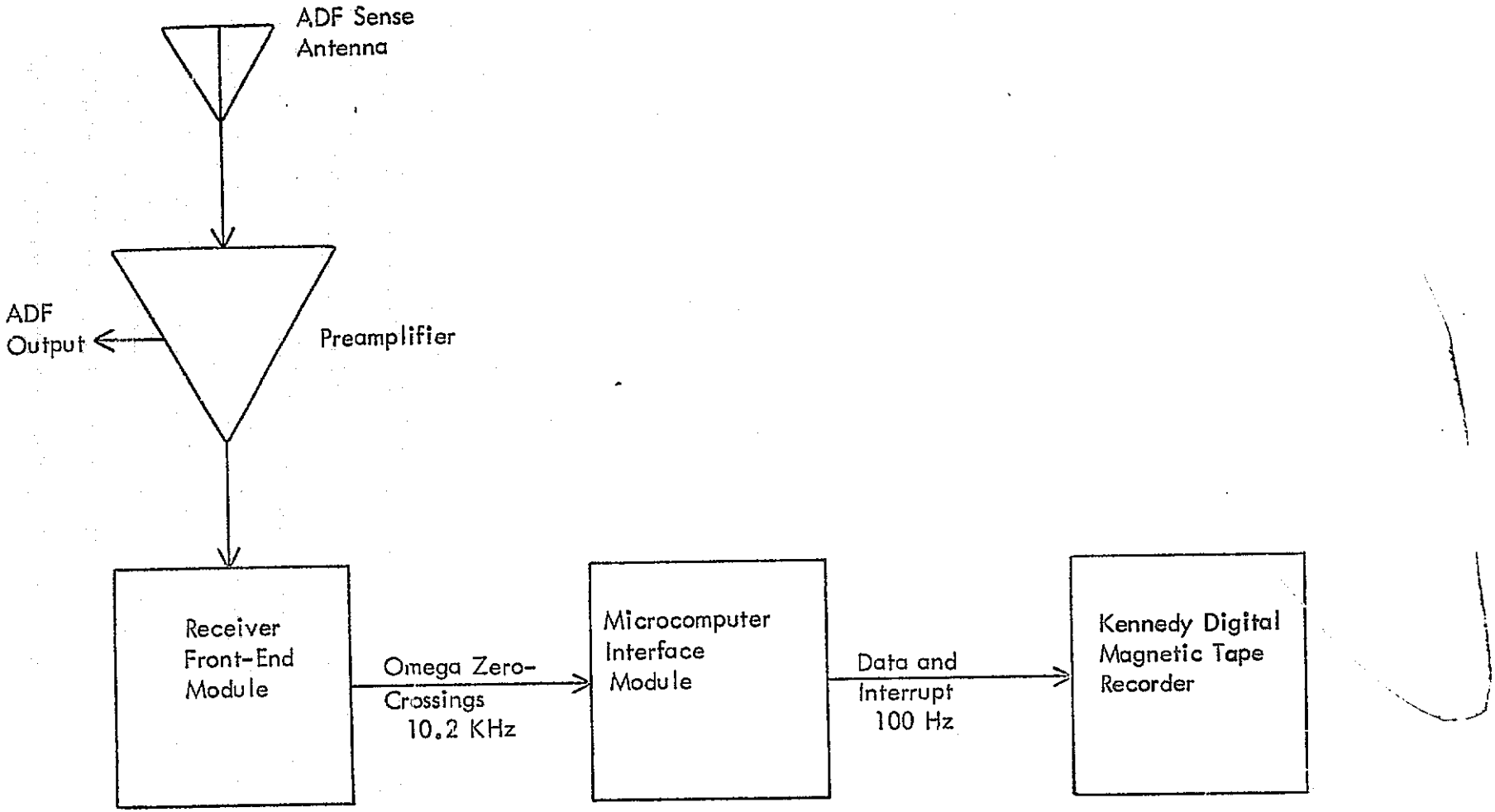


Figure B-1. Omega Data Collection Setup.

outbound flight was conducted between the hours of 7:00 and 8:30 AM on October 20, 1976. The cruise altitude was 2438 meters and VFR conditions persisted throughout the flight. The return flight was between the hours of 6:00 and 7:00 PM on the same day and the cruise altitude was 2743 meters in VFR conditions. The aircraft's ADF sense antenna mounted on top of the fuselage was used with the preamplifier mounted on the ceiling inside the aircraft. The preamplifier fed approximately 6.1 meters of coaxial lead-in cable to the receiver.

All data was recorded on magnetic tape in the binary format presented by the microcomputer interface module. The phase of the Omega signal is recorded every ten milliseconds, each data point being quantized into the interval from zero to 255. The OSP's temperature compensated crystal oscillator (TCXO) was used as the local reference clock; therefore, the slight frequency offset of the TCXO appears as a constant-velocity phase ramp in the data. Figure B-2 summarizes the contents of the data base.

File #	Type Data	Date Collected	Location	Stations Transmitting
1	Ground (24 hours)	January, 1976	Athens, Ohio	B, C, D, G
2	Ground (D off air)	June, 1976	Athens, Ohio	B, C, G
3	Ground (D off air)	June, 1976	Athens, Ohio	B, C, G
4	Flight (VFR)	October, 1976	Athens-Chicago	B, C, D, G
5	Flight (VFR)	October, 1976	Chicago-Athens	B, C, D, G

Figure B-2. Omega Data Base Summary.

C. Omega Sensor Processor Circuitry. The Omega sensor processor is composed of three distinct hardware modules and a general-purpose microcomputer. The circuit details of the Omega preamplifier, receiver front-end, and microcomputer interface modules are given in Figures C-1, C-2 and C-3, respectively. The microcomputer system is described in Appendix D.

D. JOLT Microcomputer System.

1. Hardware. The JOLT Microcomputer System is available from Microcomputer Associates, Inc., Santa Clara, California. As implemented in the Omega Sensor Processor, the JOLT system consists of a CPU board and a memory board. The CPU board containing the 6502 CPU chip, 6530 chip (PIA, TTY Monitor ROM, and Interval Timer), 6520 PIA chip, TTY interface circuitry, and 512 bytes RAM is available in kit form for $150. The 4K RAM memory board is available in kit form for $200. Circuit diagrams for the CPU and memory boards appear in Figures D-1 and D-2.[22]

As can be seen from these diagrams, the JOLT incorporates a 16-line address bus and an 8-line data bus, along with an adjustable 700 KHz - 1 MHz oscillator. System architecture includes one accumulator, two index registers, one processor status register, one stack pointer register, and one program counter register. All are 8-bit registers except for the program counter which is 16-bits long.

The system accommodates two fully vectored interrupts. One is a non-maskable interrupt (NMI), and the other is an interrupt request (IRQ) which can be disabled and thereafter ignored by the software.

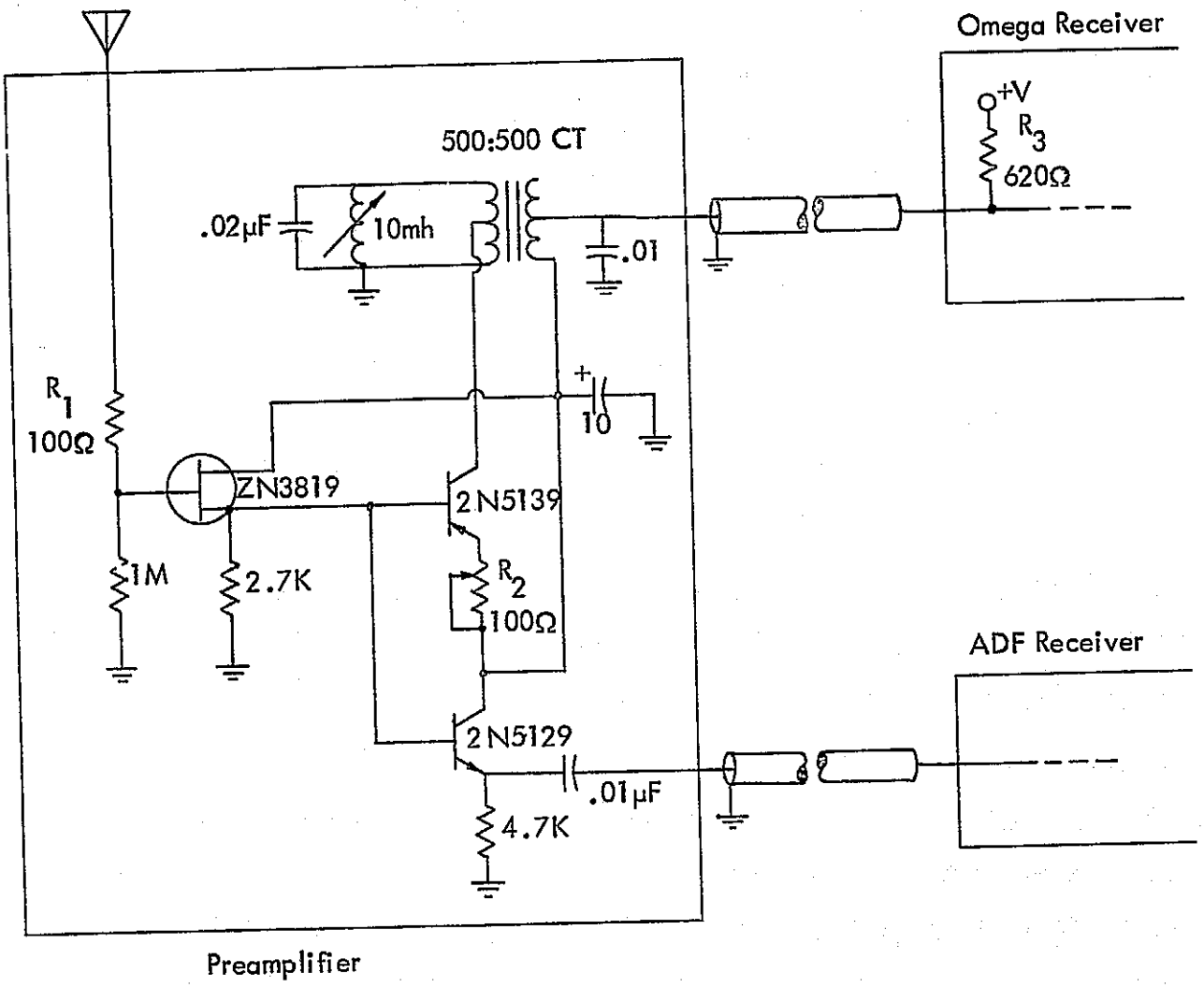


Figure C-1. Dual Purpose Preamplifier Module.

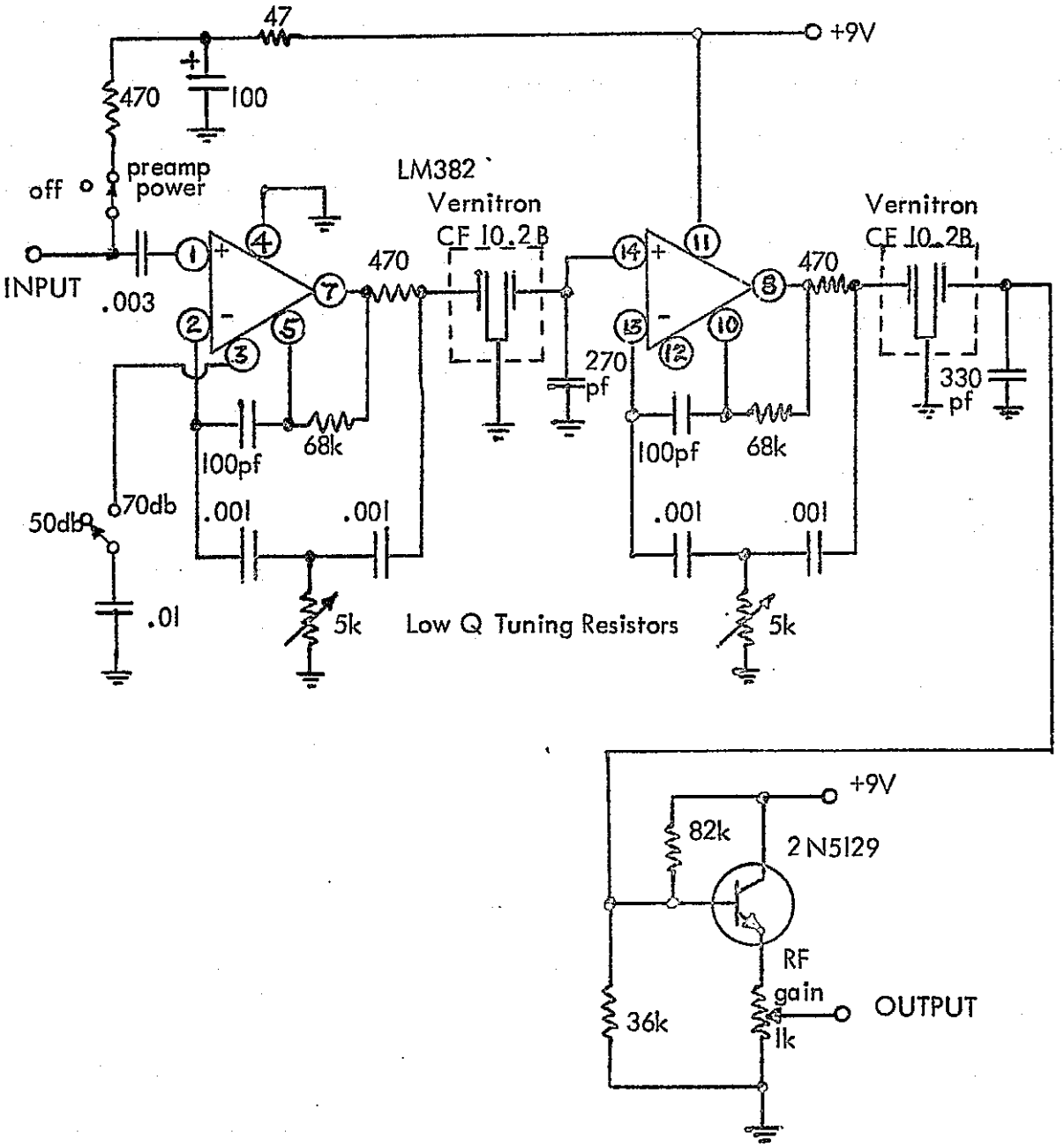


Figure C-2a. Receiver Front-End Module, Two-Stage VLF Bandpass Amplifier.

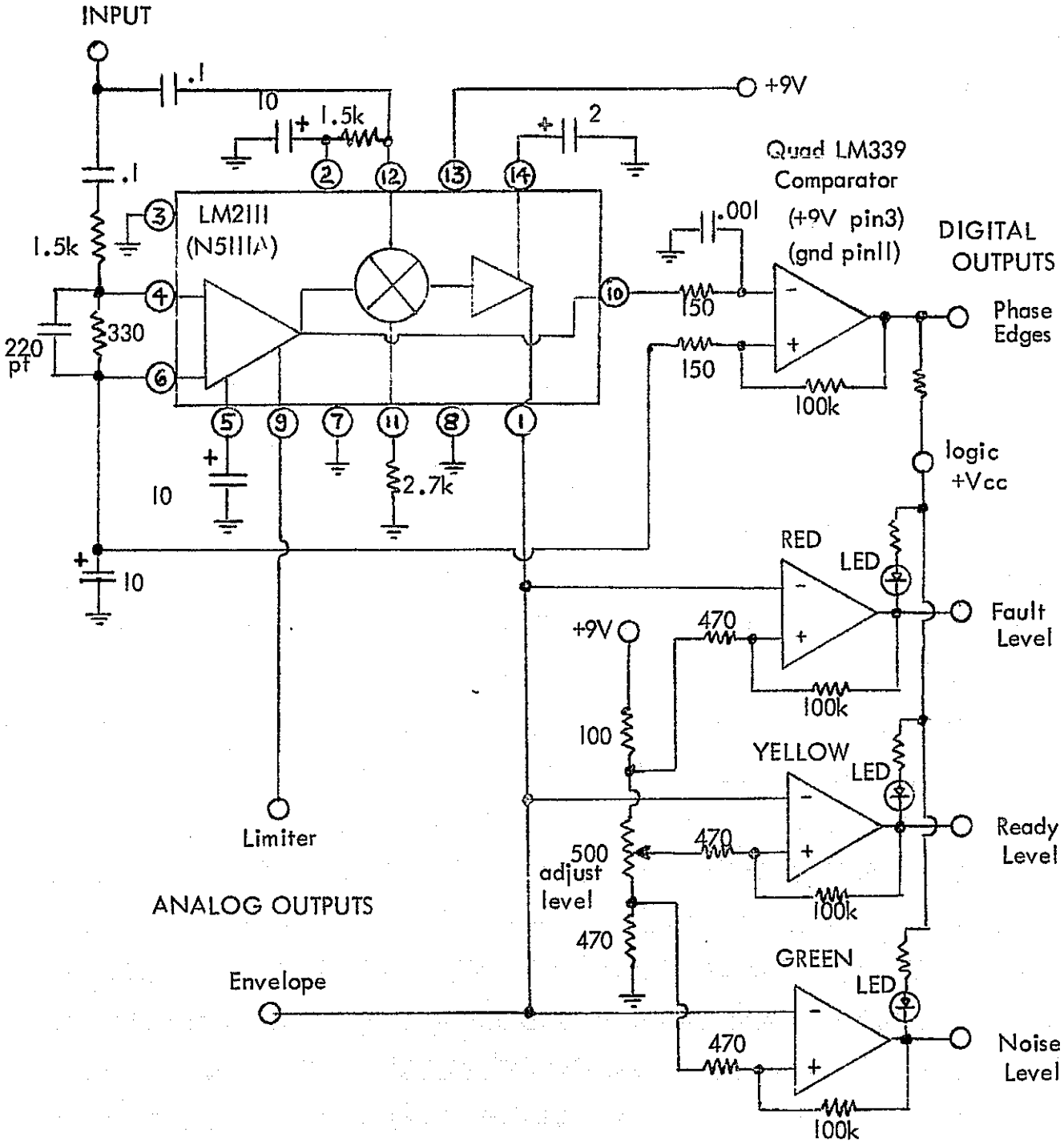


Figure C-2b. Receiver Front-End Module, Limiter - Comparator Circuit.

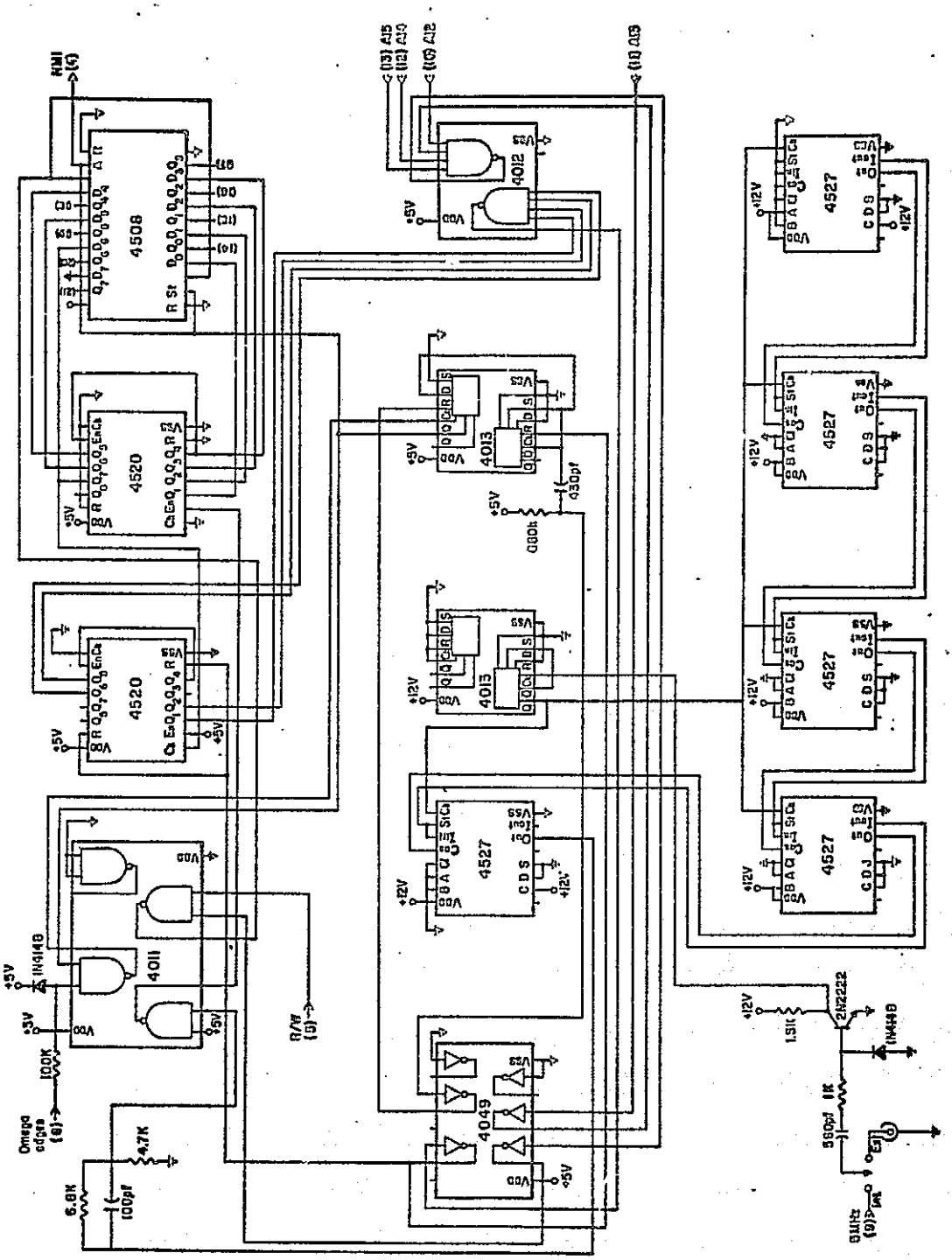
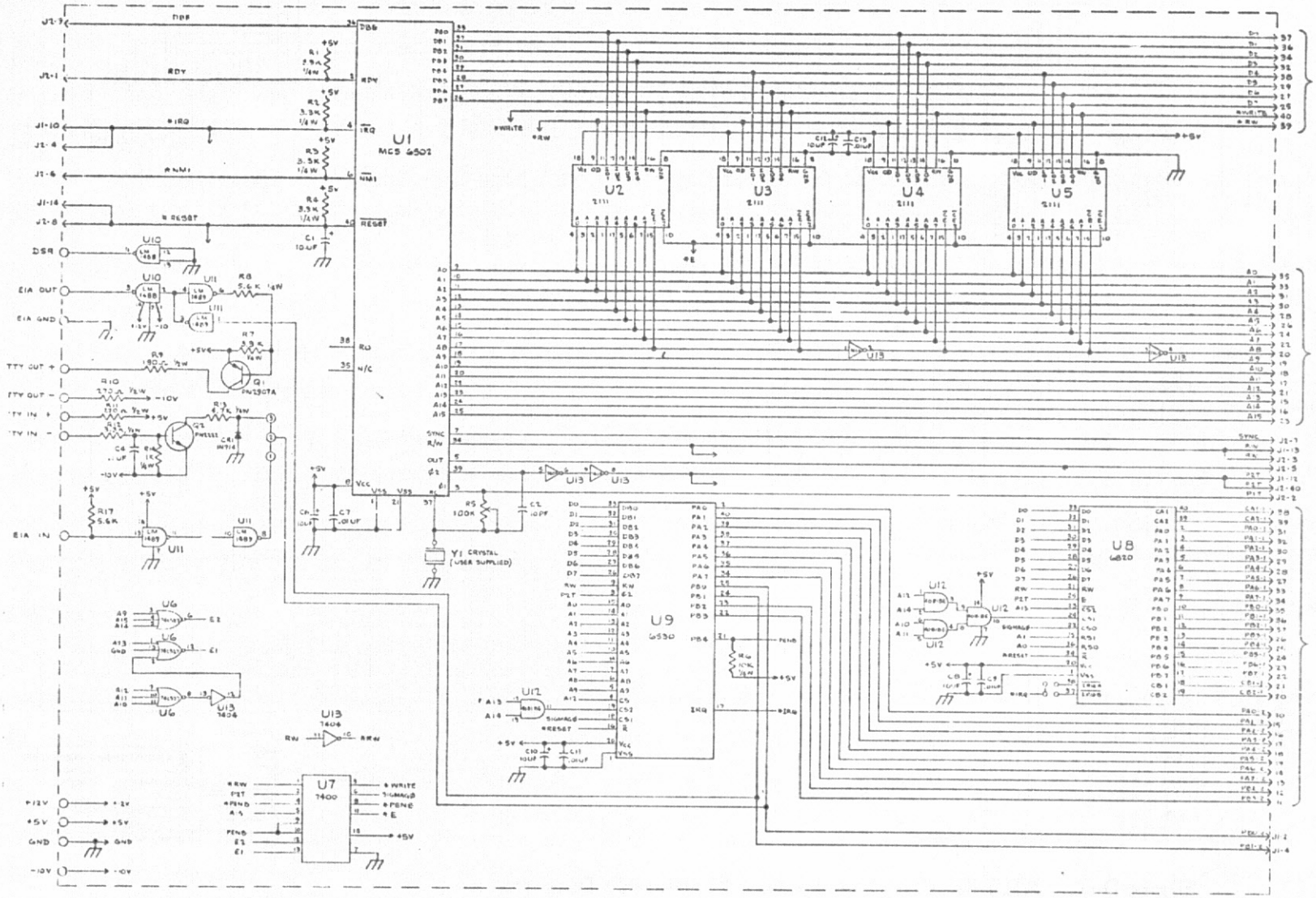


Figure C-3. Microcomputer Interface Module Circuit Diagram.

ORIGINAL PAGE IS
OF POOR QUALITY



© 1979 BY MICROCOMPUTER ASSOCIATES, INC
- ALL RIGHTS RESERVED -

Microcomputer Associates Inc

SCHEMATIC DIAGRAM,
JOLT - CPU

SIZE D | DRAWING NO. JLT-001

DO NOT SCALE DRAWING

Figure D-1. JOLT CPU Board Schematic Diagram.

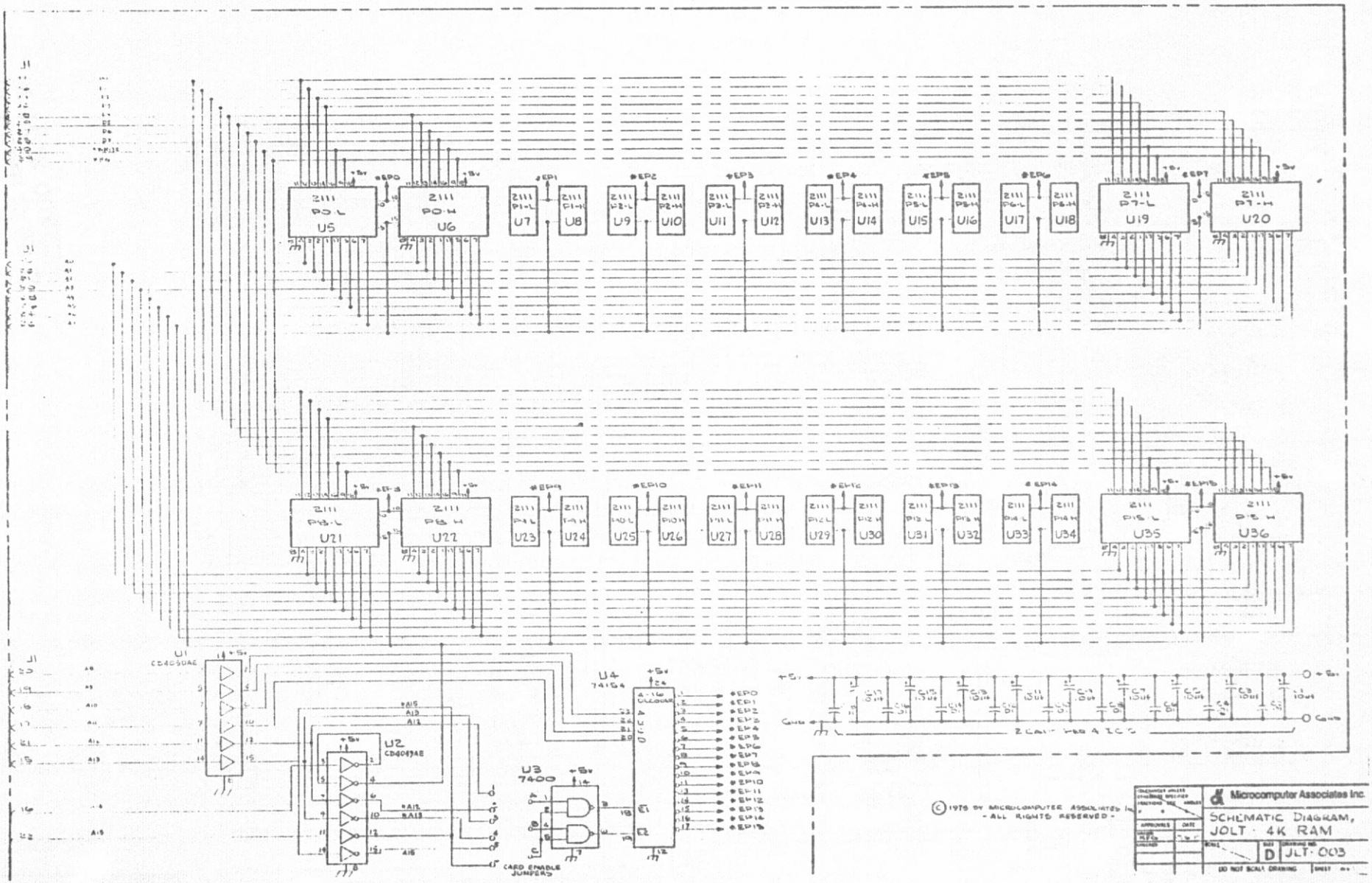


Figure D-2. JOLT 4K RAM Memory Board Schematic Diagram.

The JOLT system requires approximately 2 amps at + 5 volts and 50 milliamps at -10 volts DC. After these supplies have been turned on, the user need only push the system reset button and the system is ready to accept user-commands from a TTY keyboard.

2. Software. Figure D-3 gives the instruction set for the JOLT system and details about each instruction. It can be seen that the average instruction fetch and execute time is about 3 microseconds. The 52 instructions combined with their various addressing modes provide one hundred and fifty unique operations. Figure D-4 gives more software details and Figure D-5 gives the JOLT system memory map.[22]

The TTY monitor routine which is resident on the 6530 1K ROM chip allows the user to operate the JOLT from a TTY keyboard. Commands include: (1) display contents of memory; (2) display contents of CPU registers; (3) alter contents of memory or CPU registers; (4) begin program execution; and (5) punch and read paper tape. In addition, several monitor subroutines can be called to type data on the TTY printer.

In addition, a 6502 Assembler which can be executed via a larger computer system employing FORTRAN IV, has been obtained.[23] This Assembler capability allows the programmer to write microcomputer programs in mnemonics rather than hexadecimal number codes, and allows for use of variable names and labels rather than numerical codes and locations.

E. FORTRAN Simulation Analysis. In order to test the feasibility of performing OSP functions of automatic synchronization and PLL phase tracking in

INSTRUCTIONS		IMMEDIATE			ABSOLUTE			ZERO PAGE			ACCUM.			IMPLIED			(IND,X)			(IND),Y			Z,PAGE,X			ABS,X			ABS,Y			RELATIVE			INDIRECT			Z,PAGE,Y			CONDITION CODES					
MNEMONIC	OPERATION	OP	N	#	OP	N	#	OP	N	#	OP	N	#	OP	N	#	OP	N	#	OP	N	#	OP	N	#	OP	N	#	OP	N	#	OP	N	#	OP	N	#	OP	N	#	N	Z	C	I	D	V
ADC	A+M+C → A	(1)	69	2	2	6D	4	3	65	3	2						61	6	2	71	5	2	75	4	2	7D	4	3	79	4	3											✓	✓	✓	-	-	✓
AND	A∧M → A	(1)	29	2	2	2D	4	3	25	3	2						21	6	2	31	5	2	35	4	2	3D	4	3	39	4	3											✓	✓	-	-	-	-
ASL	C←[7…0]←0							0E	6	3	0A	2	1										16	6	2	1E	7	3													✓	✓	✓	-	-	-
BCC	BRANCH ON C=0	(2)																																	90	2	2											
BCS	BRANCH ON C=1	(2)																																	B0	2	2											
BEQ	BRANCH ON Z=1	(2)																																	F0	2	2											
BIT	A∧M							2C	4	3	24	3	2																												M7	✓	-	-	-	M6
BMI	BRANCH ON N=1	(2)																																	30	2	2											
BNE	BRANCH ON Z=0	(2)																																	D0	2	2											
BPL	BRANCH ON N=0	(2)																																	10	2	2											
BRK	(See Fig 1)													00	7	1																									-	-	-	1	-	-
BVC	BRANCH ON V=0	(2)																																	50	2	2											
BVS	BRANCH ON V=1	(2)																																	70	2	2											
CLC	0→C													18	2	1																									-	-	0	-	-	-
CLD	0→D													D8	2	1																									-	-	-	-	0	-
CLI	0→I													58	2	1																									-	-	-	0	-	-
CLV	0→V													B8	2	1																									-	-	-	-	-	0
CMP	A-M	(1)	C9	2	2	CD	4	3	C5	3	2						C1	6	2	D1	5	2	D5	4	2	DD	4	3	D9	4	3											✓	✓	✓	-	-	-
CPX	X-M		E0	2	2	EC	4	3	E4	3	2																														✓	✓	✓	-	-	-
CPY	Y-M		C0	2	2	CC	4	3	C4	3	2																														✓	✓	✓	-	-	-
DEC	M-1→M					CE	6	3	C6	5	2												D6	6	2	DE	7	3													✓	✓	-	-	-	-
DEX	X-1→X													CA	2	1																									✓	✓	-	-	-	-
DEY	Y-1→Y													88	2	1																									✓	✓	-	-	-	-
EOR	A⊻M→A	(1)	49	2	2	4D	4	3	45	3	2						41	6	2	51	5	2	55	4	2	5D	4	3	59	4	3											✓	✓	-	-	-	-
INC	M+1→M					EE	6	3	E6	5	2												F6	6	2	FE	7	3													✓	✓	-	-	-	-
INX	X+1→X													E8	2	1																									✓	✓	-	-	-	-
INY	Y+1→Y													C8	2	1																									✓	✓	-	-	-	-
JMP	JUMP TO NEW LOC					4C	3	3																								6C	5	3											
JSR	(See Fig 2) JSR SUB					20	6	3																																	-	-	-	-	-	-
LDA	M→A		A9	2	2	AD	4	3	A5	3	2						A1	6	2	B1	5	2	B5	4	2	BD	4	3	B9	4	3											✓	✓	-	-	-	-

MNEMONIC	OPERATION	IMMEDIATE			ABSOLUTE			ZERO PAGE			ACCUM.			IMPLIED			(IND,X)			(IND),Y			Z,PAGE,X			ABS,X			ABS,Y			RELATIVE			INDIRECT			Z,PAGE,Y			CONDITION CODES					
		OP	N	#	OP	N	#	OP	N	#	OP	N	#	OP	N	#	OP	N	#	OP	N	#	OP	N	#	OP	N	#	OP	N	#	OP	N	#	OP	N	#	OP	N	#	N	Z	C	I	D	V
LDX	M→X	(1)	A2	2	2	AE	4	3	A6	3	2																		BE	4	3											✓	✓	-	-	-	-
LDY	M→Y	(1)	A0	2	2	AC	4	3	A4	3	2												94	4	2	BC	4	3													✓	✓	-	-	-	-
LSR	0→[7…0]→C					4E	6	3	46	5	2	4A	2	1									56	6	2	5E	7	3													0	✓	✓	-	-	-
NOP	NO OPERATION													EA	2	1																									-	-	-	-	-	-
ORA	A∨M→A		09	2	2	0D	4	3	05	3	2						01	6	2	11	5	2	15	4	2	1D	4	3	19	4	3											✓	✓	-	-	-	-
PHA	A→Ms S-1→S													48	3	1																									-	-	-	-	-	-
PHP	P→Ms S-1→S													08	3	1																									-	-	-	-	-	-
PLA	S+1→S Ms→A													68	4	1																									✓	✓	-	-	-	-
PLP	S+1→S Ms→P													28	4	1																									(RESTORED)					
ROL	C←[7…0]←C					2E	6	3	26	5	2	2A	2	1									36	6	2	3E	7	3													✓	✓	✓	-	-	-
ROR	C→[7…0]→C					6E	6	3	66	5	2	6A	2	1									76	6	2	7E	7	3													✓	✓	✓	-	-	-
RTI	(See Fig. 3) RTI FROM INT													40	6	1																									(RESTORED)					
RTS	(See Fig. 2) RTRN SUB													60	6	1																									-	-	-	-	-	-
SBC	A-M-C→A	(1)	E9	2	2	ED	4	3	E5	3	2						E1	6	2	F1	5	2	F5	4	2	FD	4	3	F9	4	3											✓	✓	✓	-	-	✓
SEC	1→C													38	2	1																									-	-	1	-	-	-
SED	1→D													F8	2	1																									-	-	-	-	1	-
SEI	1→I													78	2	1																									-	-	-	1	-	-
STA	A→M					8D	4	3	85	3	2						81	6	2	91	6	2	95	4	2	9D	5	3	99	5	3											-	-	-	-	-	-
STX	X→M					8E	4	3	86	3	2																															-	-	-	-	-	-
STY	Y→M					8C	4	3	84	3	2												94	4	2																	-	-	-	-	-	-
TAX	A→X													AA	2	1																									✓	✓	-	-	-	-
TAY	A→Y													A8	2	1																									✓	✓	-	-	-	-
TSX	S→X													BA	2	1																									✓	✓	-	-	-	-
TXA	X→A													8A	2	1																									✓	✓	-	-	-	-
TXS	X→S													9A	2	1																									-	-	-	-	-	-
TYA	Y→A													98	2	1																									✓	✓	-	-	-	-

(1) ADD 1 TO "N" IF PAGE BOUNDRY IS CROSSED
(2) ADD 1 TO "N" IF BRANCH OCCURS TO SAME PAGE
ADD 2 TO "N" IF BRANCH OCCURS TO DIFFERENT PAGE.
(3) CARRY NOT = BORROW
(4) IN DECIMAL MODE Z FLAG IS INVALID.
ACCUMULATOR MUST BE CHECKED FOR ZERO RESULT

X INDEX - X
Y INDEX - Y
A ACCUMULATOR
M MEMORY PER EFFECTIVE ADDRESS
Ms MEMORY PER STACK POINTER

+ ADD
- SUBTRACT
∧ AND
∨ OR
⊻ EXCLUSIVE OR
✓ MODIFIED
- NOT MODIFIED
M7 MEMORY BIT 7
M6 MEMORY BIT 6

N NO. CYCLES
NO. BYTES

OP-CODE TABLE

LSD / MSD	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	LSD / MSD
0	BRK	ORA-IND,X				ORA-Z,Page	ASL-Z,PAGE		PHP	ORA-IMM	ASL-A			ORA-ABS	ASL-ABS		0
1	BPL	ORA-IND,Y				ORA-Z,Page,X	ASL-Z,Page,X		CLC	ORA-ABS,Y				ORA-ABS,X	ASL-ABS,X		1
2	JSR	AND-IND,X			BIT-Z Page	AND-Z, Page	ROL-Z,Page		PLP	AND-IMM	ROL-A		BIT-ABS	AND-ABS	ROL-ABS		2
3	BMI	AND-IND,Y				AND-Z,Page,X	ROL-Z,Page,X		SEC	AND-ABS,Y				AND-ABS,X	ROL-ABS,X		3
4	RTI	EOR-IND,X				EOR-Z,Page	LSR-Z,Page		PHA	EOR-IMM	LSR-A		JMP-ABS	EOR-ABS	LSR-ABS		4
5	BVC	EOR-IND,Y				EOR-Z,Page,X	LSR-Z,Page,X		CLI	EOR-ABS,Y				EOR-ABS,X	LSR-ABS,Y		5
6	RTS	ADC-IND,X				ADC-Z, Page			PLA	ADC-IMM			JMP-IND	ADC-ABS			6
7	BVS	ADC-IND,Y				ADC-Z,Page			SEI	ADC-ABS,Y				ADC-ABS,X			7
8		STA-IND,X			STY-Z, Page	STA-Z, Page	STX-Z, Page		DEY		TXA		STY-ABS	STA-ABS	STX-ABS		8
9	BCC	STA-IND,Y			STY-Z,Page,X	STA-Z,Page,X	STX-Z,Page,Y		TYA	STA-ABS,Y	TXS			STA-ABS,X			9
A	LDY-IMM	LDA-IND,X	LDX-IMM		LDY-Z Page	LDA-Z Page	LDX-Z, Page		TAY	LDA-IMM	TAX		LDY-ABS	LDA-ABS	LDX-ABS		A
B	BCS	LDA-IND,Y			LDY-Z,Page,X	LDA-Z,Page,X	LDX-Z,Page,Y		CLV	LDA-ABS,Y	TSX		LDY-ABS,X	LDA-ABS,X	LDX-ABS,Y		B
C	CPY-IMM	CMP-IND,X			CPY-Z Page	CMP-Z, Page	DEC-Z Page		INY	CMP-IMM	DEX		CPY-ABS	CMP-ABS	DEC-ABS		C
D	BNE	CMP-IND,Y				CMP-Z,Page,X	DEC-Z,Page,X		CLD	CMP-ABS,Y				CMP-ABS,X	DEC-ABS,X		D
E	CPX-IMM	SBC-IND,X			CPX-Z, Page	SBC-Z, Page	INC-Z, Page		INX	SBC-IMM	NOP		CPX-ABS	SBC-ABS	INC-ABS		E
F	BEQ	SBC-IND,Y				SBC-Z,Page,X	INC-Z,Page,X		SED	SBC-ABS,Y				SBC-ABS,X	INC-ABS,X		F

IMM — IMMEDIATE ADDRESSING – THE OPERAND IS CONTAINED IN THE SECOND BYTE OF THE INSTRUCTION

ABS — ABSOLUTE ADDRESSING – THE SECOND BYTE OF THE INSTRUCTION CONTAINS THE 8 LOW ORDER BITS OF THE EFFECTIVE ADDRESS. THE THIRD BYTE CONTAINS THE 8 HIGH ORDER BITS OF THE EFFECTIVE ADDRESS

Z PAGE ZERO PAGE ADDRESSING – SECOND BYTE CONTAINS THE 8 LOW ORDER BITS OF THE EFFECTIVE ADDRESS. THE 8 HIGH ORDER BITS ARE ZERO.

A — ACCUMULATOR – ONE BYTE INSTRUCTION OPERATING ON THE ACCUMULATOR

Z PAGE X, Z PAGE Y ZERO PAGE INDEXED – THE SECOND BYTE OF THE INSTRUCTION IS ADDED TO THE INDEX (CARRY IS DROPPED) TO FORM THE LOW ORDER BYTE OF THE EA. THE HIGH ORDER BYTE OF THE EA IS ZEROS

ABS,X ABS,Y ABSOLUTE INDEXED: THE EFFECTIVE ADDRESS IS FORMED BY ADDING THE INDEX TO THE SECOND AND THIRD BYTE OF THE INSTRUCTION

(IND, X) INDEXED INDIRECT – THE SECOND BYTE OF THE INSTRUCTION IS ADDED TO THE X INDEX, DISCARDING THE CARRY. THE RESULTS POINTS TO A LOCATION ON PAGE ZERO WHICH CONTAINS THE 8 LOW ORDER BITS OF THE EA. THE NEXT BYTE CONTAINS THE 8 HIGH ORDER BITS

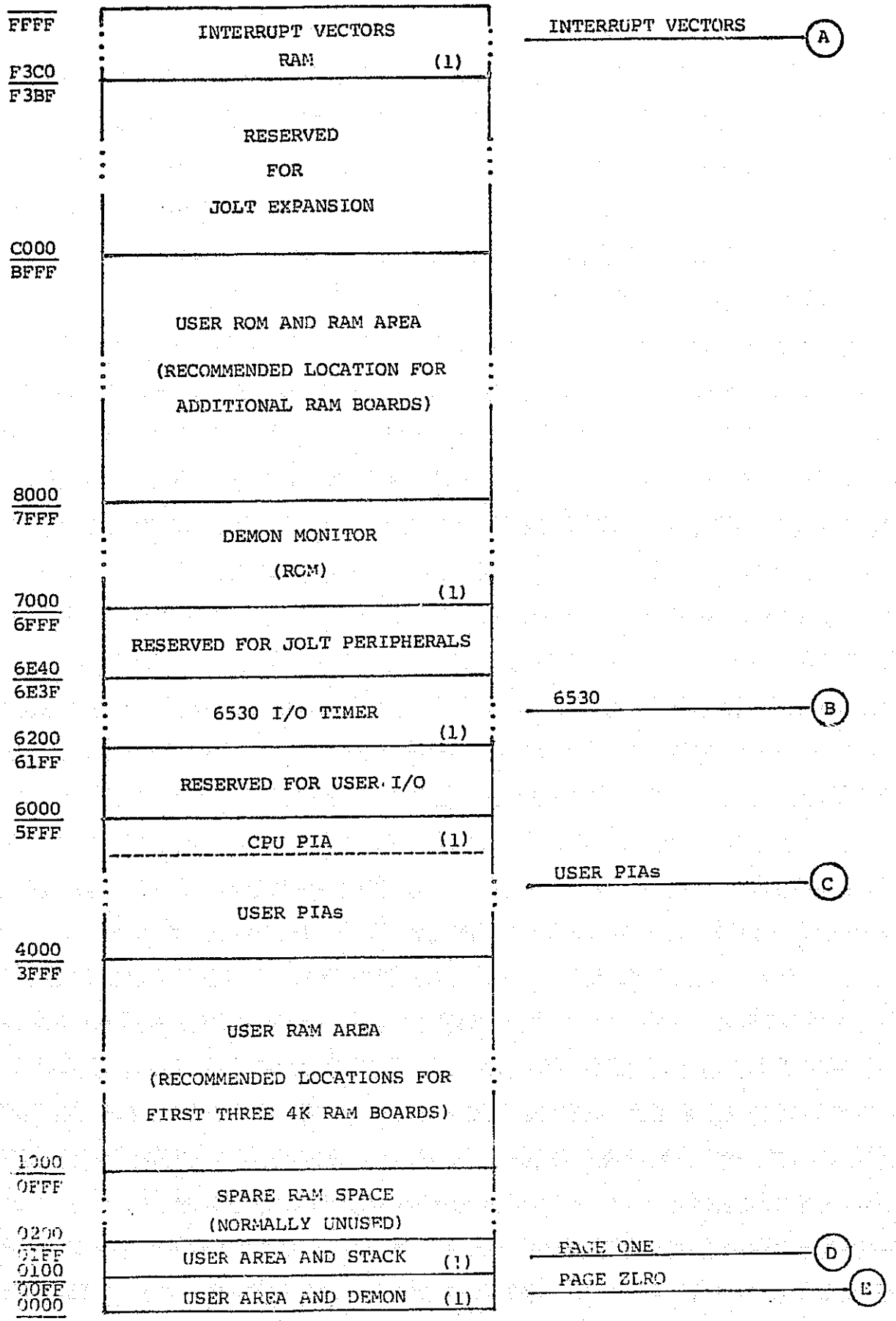
(IND),Y INDIRECT INDEXED – THE SECOND BYTE OF THE INSTRUCTION POINTS TO A LOCATION IN PAGE ZERO. THE CONTENTS OF THIS MEMORY LOCATION IS ADDED TO THE Y INDEX, THE RESULT BEING THE LOW ORDER EIGHT BITS OF THE EA. THE CARRY FROM THIS OPERATION IS ADDED TO THE CONTENTS OF THE NEXT PAGE ZERO LOCATION, THE RESULT BEING THE 8 HIGH ORDER BITS OF THE EA

Figure D-3. Instruction Set for MOS Technology 6502 CPU.

ORIGINAL PAGE IS OF POOR QUALITY

COMMAND	OPERATION
R	Print out the CPU registers.
M	Print out contents of memory.
:	Alter contents of CPU registers or memory.
G	Begin program execution.
WH	Write contents of memory in hexadecimal.
LH	Read punched paper tape.

Figure D-4. JOLT System TTY Monitor Commands Summary.



(1) Standard on JOLT CPU board.

Figure D-5. JOLT System Memory Map.

software, a FORTRAN simulation analysis was performed. This simulation also served to optimize parameters in both algorithms before they were assembled in microcomputer code. The input data for much of the simulation analysis was the real Omega data base described in Appendix B; however, since the specification of system performance requires complete and absolute knowledge of the input data, computer generated data was also used.

Each simulation is summarily described, and the actual listing for each FORTRAN simulation appears at the end of this Appendix.

1. Noise Modeling. Since the real Omega data base contained uncertainties such as signal-to-noise ratios (SNR), phase measurements designed to approximate the data at the output of the OSP front-end and microcomputer interface modules were generated with the computer. The Omega noise is both gaussian and impulsive. Gaussian noise is easily generated by the computer and was judged to sufficiently approximate real Omega noise following comparisons of statistics of the real and modeled noise.

The noise model must simulate the gaussian distributed input noise being bandpassed through the 30 Hz narrowband Omega receiver front-end module. The basis for such a model of a sinusoidal signal embedded in gaussian noise and received through a narrowband receiver is developed by Schwartz. [24]

The signal developed across the antenna can be expressed as: $s(t) = (x+A)\cos(wt+o)+(y)\sin(wt)$. This expression neglects possible interferring carrier terms from other Omega frequencies and other strong signals near the desired frequency. $A\cos(wt+o)$ is the desired signal term, and x and y are the white, gaussian noise terms (with x and y independent, of zero mean, and variance σ^2).

These noise terms can be generated by calls to system library random number generator routines, RANDNR and RANDNRB. The narrowband receiver front-end module is simulated by a low-pass filter (LPF) subroutine, GRANGE. Passing the random noise terms through the LPF results in correlated noise samples. The phase angle detected at the output of the LPF front-end can be simulated by taking $\theta = \arctan(\frac{y}{x+A})$. The density function of phase obtained in this manner is:

$$f(\theta) = \frac{e^{-s^2}}{2\pi} + \frac{1}{2}\sqrt{\frac{s^2}{\pi}} \cos\theta e^{-s^2\sin^2\theta} [1 + \text{erf}(s \cos\theta)]$$

where s^2 is the power signal-to-noise ratio equal to $\frac{A^2}{2\sigma^2}$.

Comparison of the histograms and autocorrelation plots in Figures E-1, E-2 versus E-3, E-4 gives an example of how well the modeled noise approximates the real Omega noise. Lack of autocorrelation of the simulated narrowband noise is due to the fact that the LPF subroutine GRANGE simulates a single-pole RC low-pass rather than the multiple-pole low-pass filter provided by the real Omega front-end. However, the amplitude of the R(O) term (noise variance) closely approximates that of the real data, and the histograms are fairly well matched (remember that the signal-to-noise ratios for the real data are only approximations). All plots were generated via calls to the system library routine MMPLOTT.

2. Automatic Synchronization Simulation. The automatic synchronization simulation is flow charted in Figure E-5.

The real Omega data base was used as the input data for this simulation, and the subroutine DATA is first called to insure a random starting point in the data base (simulating a random OSP turn-on time). After a completed 10-second frame of data is read, the data is "mirrored". Figure E-6 shows the 10-second

ORIGINAL PAGE IS
OF POOR QUALITY

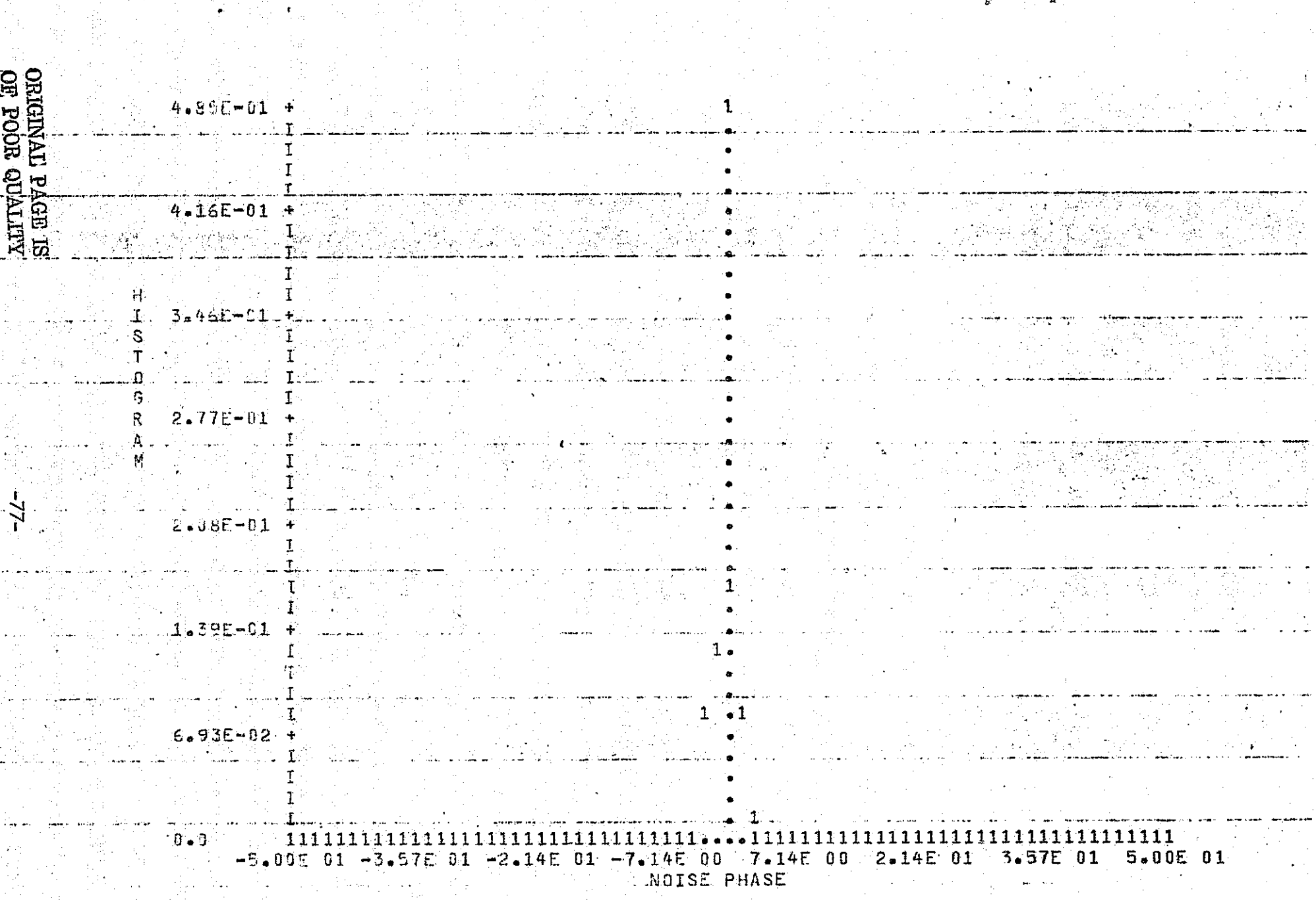


Figure E-1. Normalized Histogram of Simulated Noise Phase Vs. Phase in cec for +20 dB SNR.

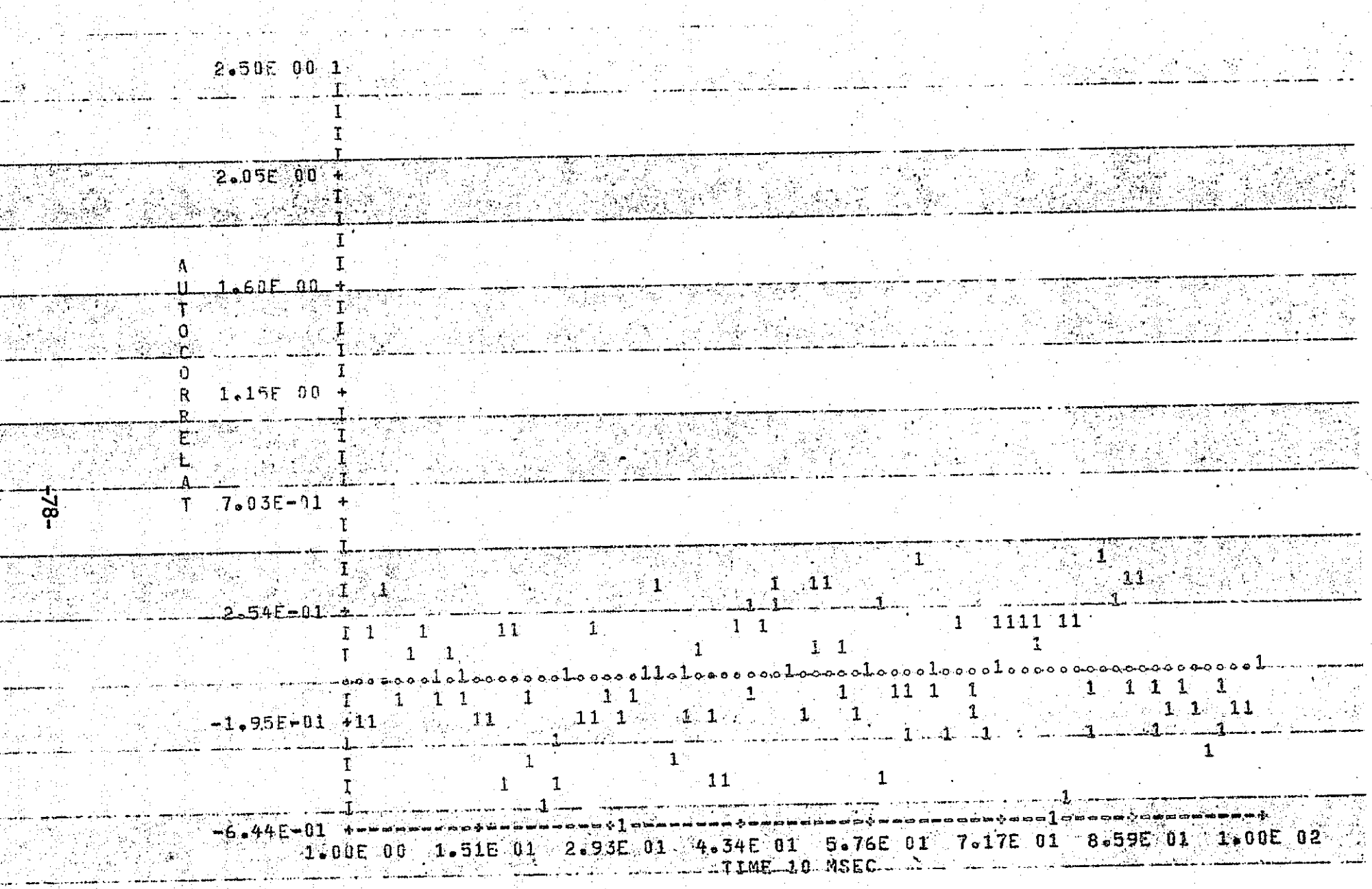


Figure E-2. Autocorrelation for Simulated Noise Phase for +20 dB SNR.

ORIGINAL PAGE IS
OF POOR QUALITY

ORIGINAL PAGE IS
OF POOR QUALITY

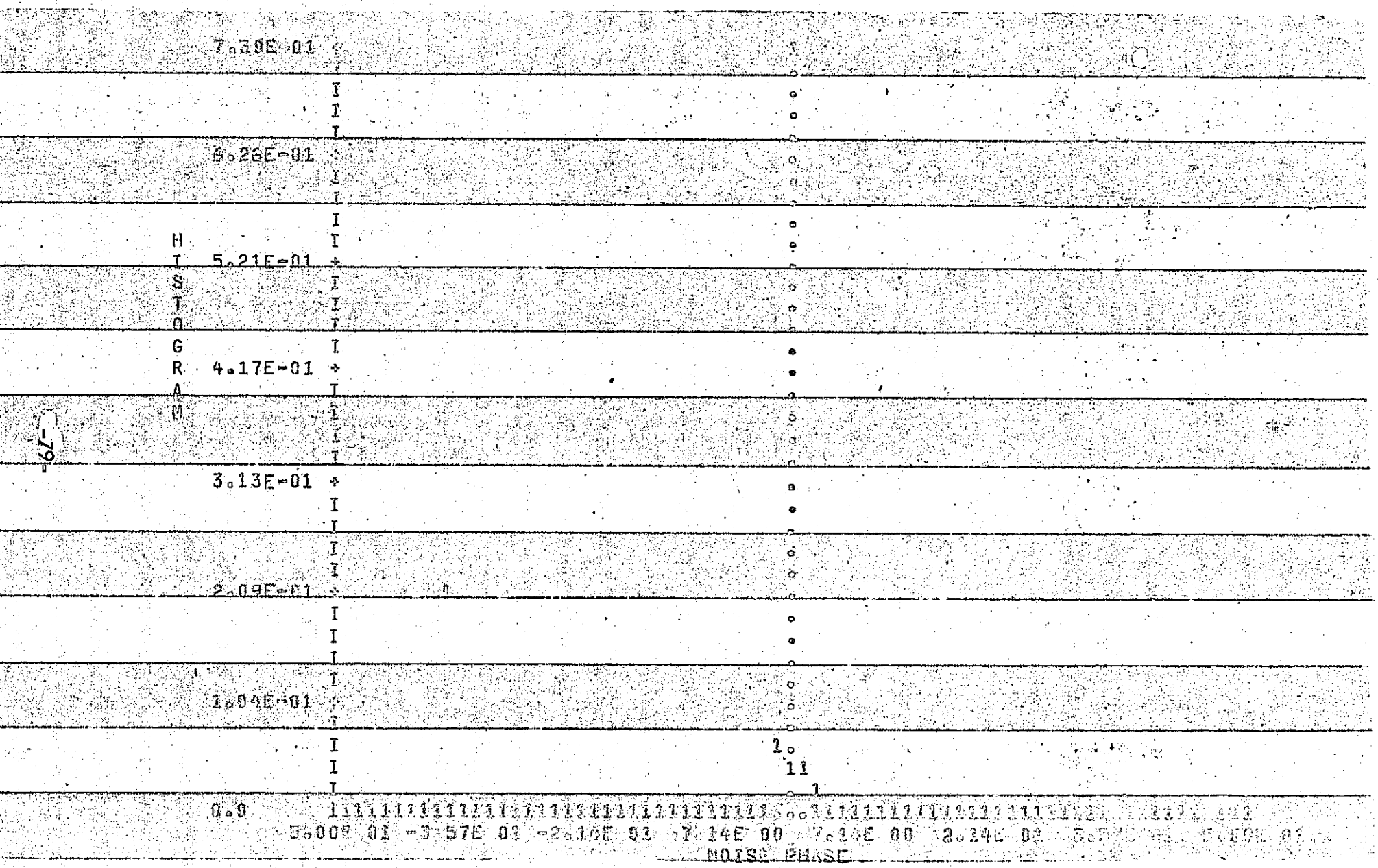


Figure E-3. Normalized Histogram of Real Omega Noise Phase Vs. Phase in cec for +20 dB SNR.

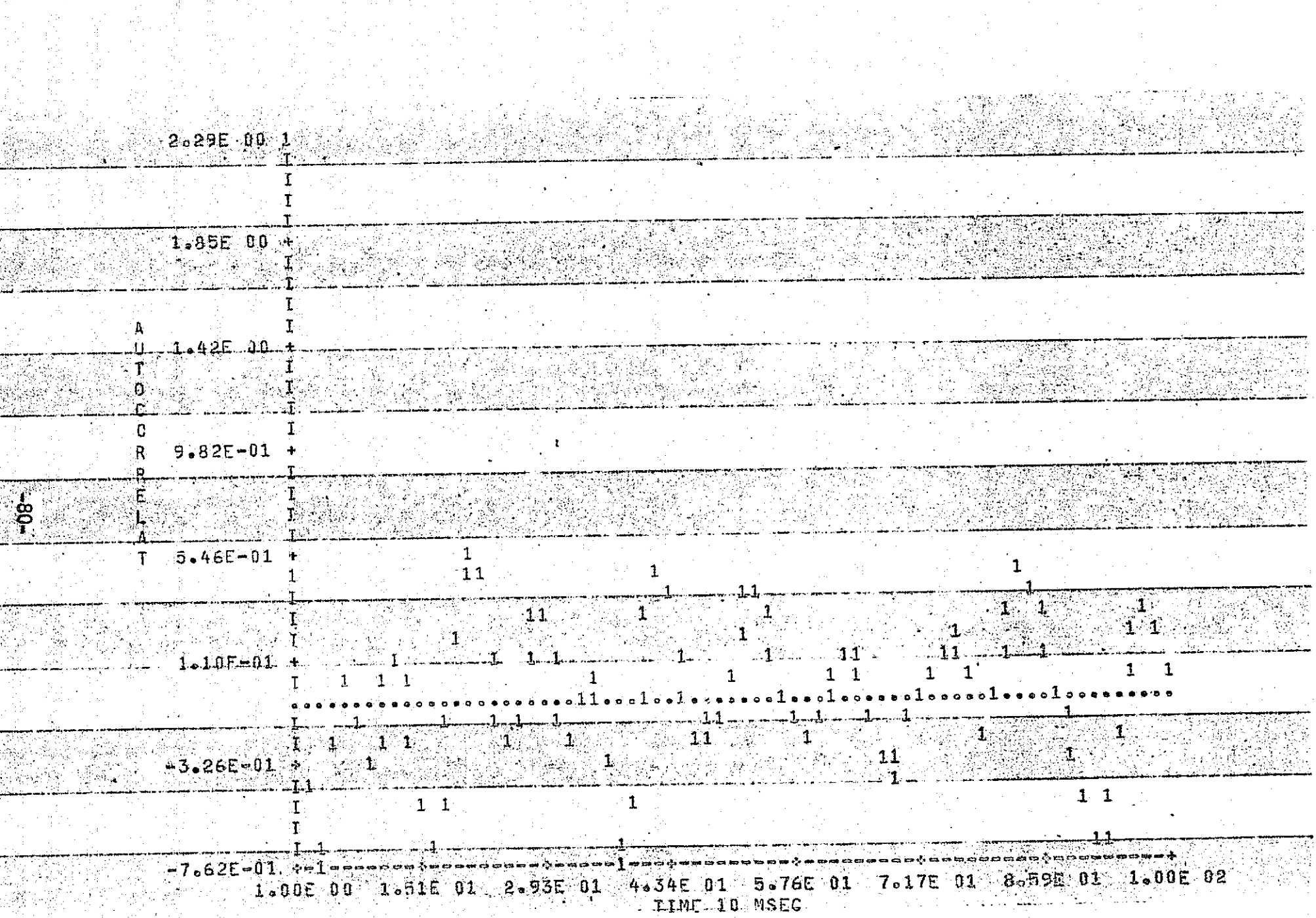


Figure E-4. Autocorrelation of Real Omega Noise Phase for +20 dB SNR.

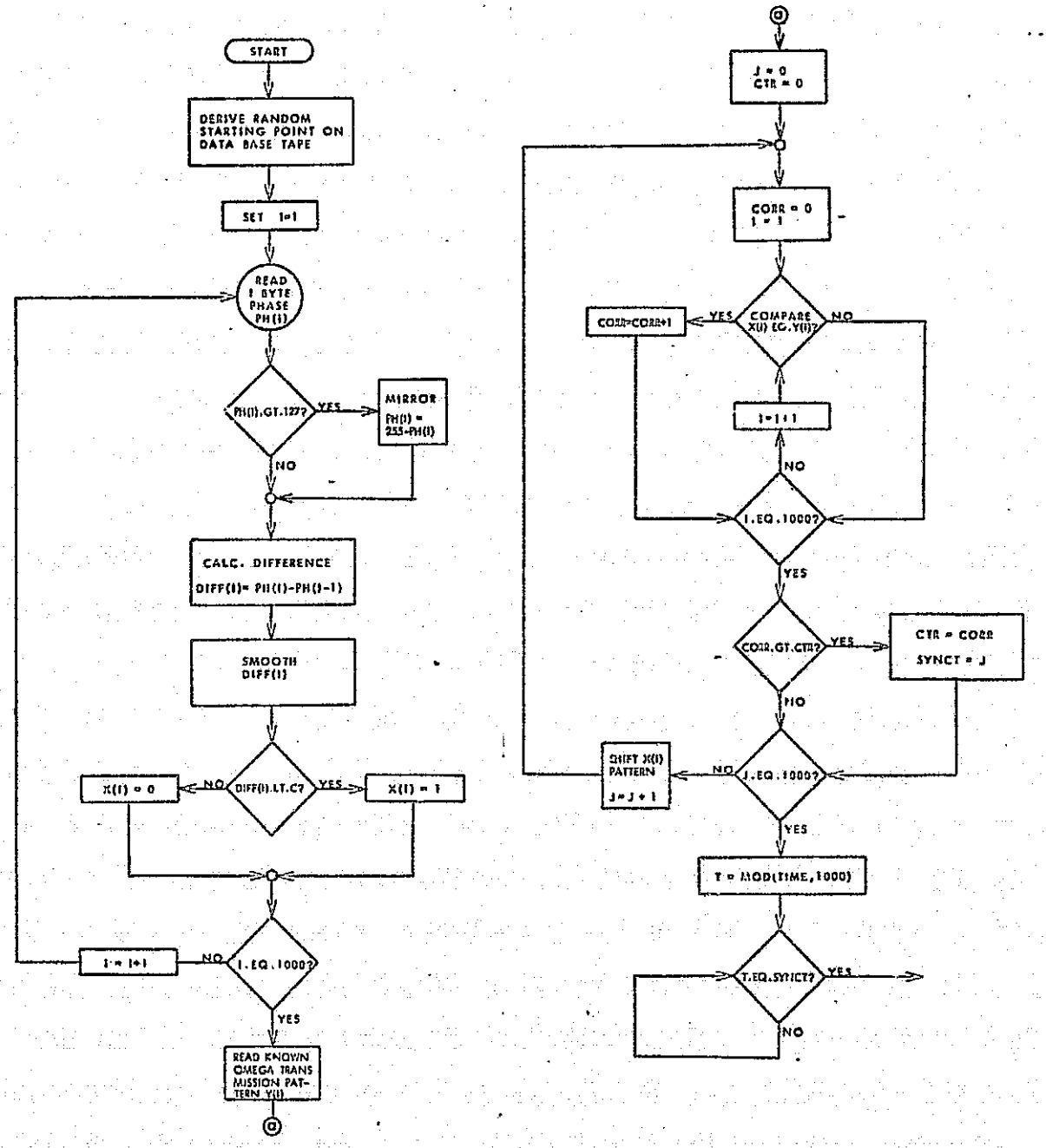


Figure E-5. Automatic Synchronization Flow Chart.

ORIGINAL PAGE IS OF POOR QUALITY

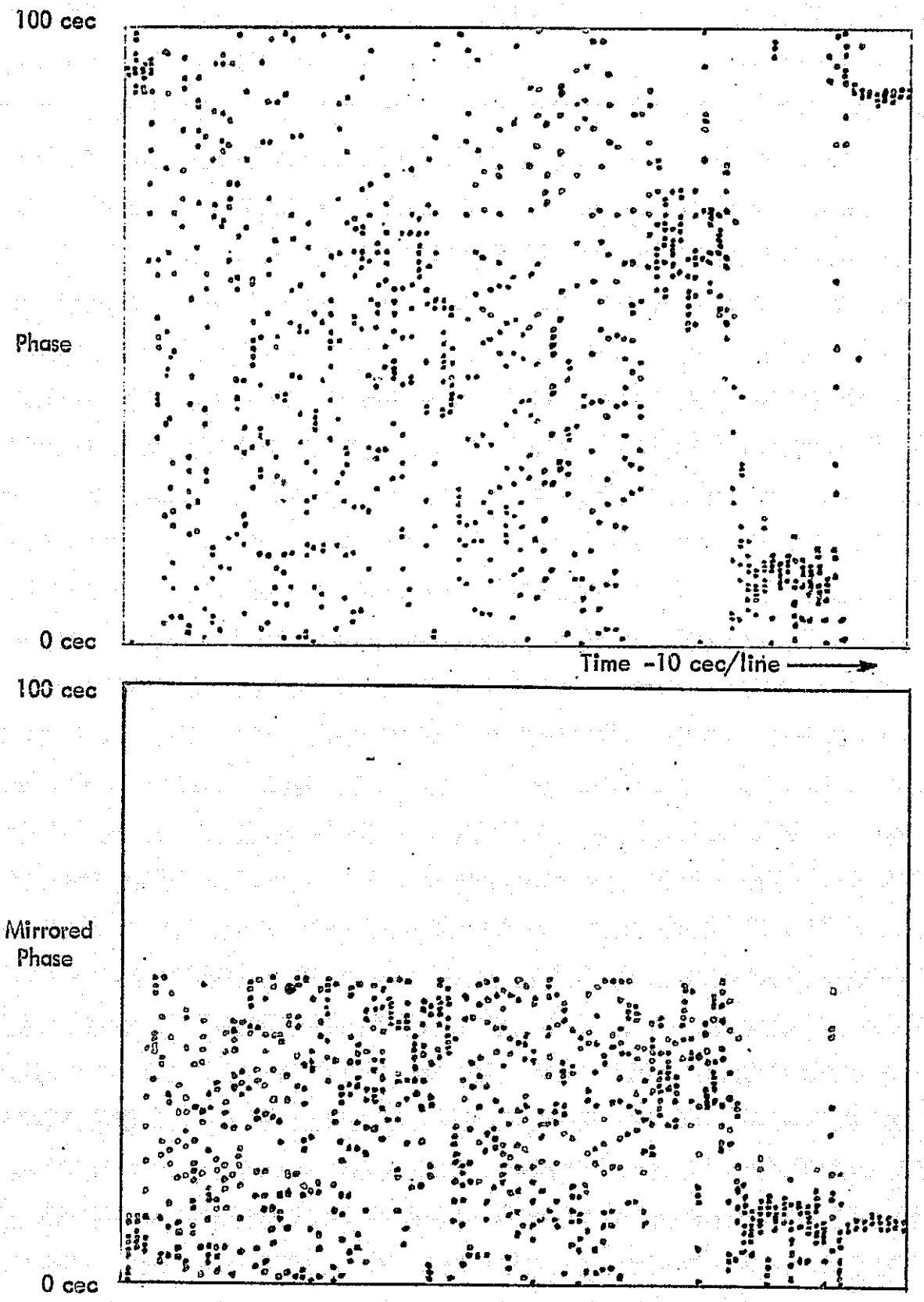


Figure E-6. 10 Second Frame of Data Plotted 10 Points Per Line.
Top = Real Data, Bottom = Mirrored Data.

frame of data and its corresponding mirrored values plotted 10 points per line. Next, sample-to-sample differences are computed (see Figure E-7). Each difference value is compared to the predetermined threshold constant, and a vector of "1" and "0" is generated which represents the correlated and uncorrelated samples.

Before generating the vector of "1" and "0" by comparing phase differences to the threshold constant, it is helpful to smooth or average these differences. Several types of smoothing filter were tried, including straight averager, one and two-sided exponential, and median three, with various amounts of points tried in each filter. The most successful was found to be the 9-point straight averager. Using this filter, the accumulated sum of nine differences is compared to nine times the threshold constant.

The threshold constant is most critical and has been determined (approximately) using the decision theory approach as well as by simulation. The constants determined by both techniques are in close agreement, thereby lending credence to the value chosen.

Having determined this constant, the vector of "1" and "0" is created by comparing each phase difference to it. Then the vector represents the received correlation pattern. This received correlation pattern can be compared (or cross-correlated) "bit-by-bit" in a bilevel correlation process with the stored unique Omega transmission pattern. A correlation counter is incremented for each coincidence of the stored and received patterns. Then the received pattern is shifted one bit and the entire cross-correlation process is repeated until all possible starting points have been tested. The number of shifts necessary to yield the highest

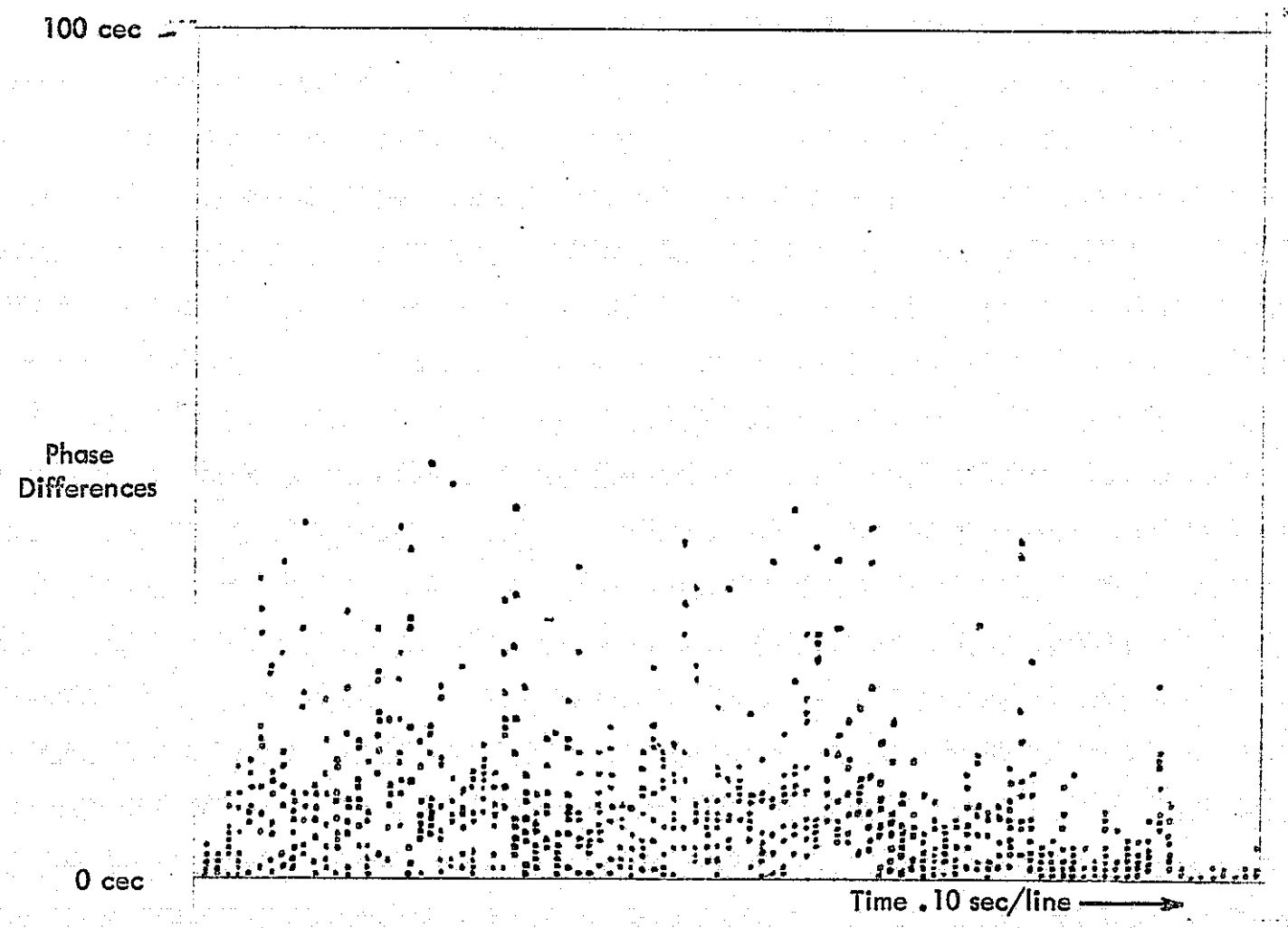


Figure E-7. Sample-to-Sample Phase Differences Plotted Ten Points Per Line.

correlation counter value is assumed to be the desired starting point for the "A" time-slot of the Omega data frame. After shifting the original data the determined number shifts, Figure 9 (in text) shows the Omega phase samples plotted ten points per line.

The routine's performance is improved if the operator inserts the stations which are expected to give coherent phase data samples (i.e. those stations received at or above 0 dB SNR). There was no noticeable difference in the routines ability to sync on ground or flight data, nor on data with the strong station North Dakota off-the-air. However, it is obvious that the technique depends on at least two stations being received above 0 dB SNR. Therefore, if only one station was sufficiently received the operator would have to identify which station it was. Note that if a two or three frequency system was available, only one station would be needed for sync (due to the unique lengths of adjacent time-slots).

3. Phase Locked Loop Simulation. Since the phase-tracking structure presented in the text (SMAPLL2) does not readily fit conventional models analyzed in the literature, computer simulation has been chosen rather than theoretical analysis to optimize loop parameters for the airborne Omega application. Computer-generated data has been used to optimize the parameters and specify SMAPLL2 performance. Finally, the real Omega data base was used to confirm expected tracking performance. Worst-case conditions of -20 dB SNR in the 30 Hz pre-detection bandwidth and 300 knots velocity of input signal phase in a direction radially to/from Omega stations have been simulated.

There are two SMAPLL2 response modes of interest: transient (including lock-up mode) and steady-state response. The transient response is evaluated by

beginning with the loop in lock with the input signal; then step and ramp input signals are applied. The response to these inputs is observed for the entire range of -20 dB to +20 dB input SNR. Steady-state response is evaluated by initializing the loop to a locked condition for various velocities of ramping input signals and calculating the loop's output error for each input signal after several hundred time-slots have been simulated. This response is also observed for the entire range of input SNR's.

A three-step approach has been utilized to "home-in" on the best choices for the parameters of first-order bandwidth and second-order gain via this simulation:

(1) With the loop in the first-order configuration (SMAPLL1), the steady-state response for a range of first-order filter constants is examined. The value of loop filtering which results in an output standard deviation of or near zero over the desired operating range of input SNR is chosen.

(2) With the loop in the second-order configuration (SMAPLL2), and using the chosen value of first-order filtering, the transient response is examined for a wide range of second-order gains. It is desirable to choose the value of second-order gain which results in best transient response (fastest lock-up time). If this minimum lock-up time is excessive, the data rate from the first-order filter is too slow, and step #1 must be repeated to choose a lower value of first-order filtering (i.e. to increase the loop's data rate).

(3) With the chosen values of first-order filtering and second-order gain, the steady-state response of the SMAPLL2 to various velocity

ramp inputs is examined. If the output error standard deviation is not within acceptable limits over the SNR operating range, steps #1 and #2 are repeated. A greater amount of first-order filtering is chosen while sacrificing some lock-up time. This trade-off is the design optimization that must be made to achieve the best combination of transient and steady-state performance for each particular application.

Executing step #1 of this approach, the SMAPLL1 yields the graph in Figure E-8. This graph gives the SMAPLL1 steady-state response to zero-velocity input signals over the desired tracking range of SNR's. Since a first-order filter of 5 bits (32 counts) (or .02 Hz bandwidth) is seen to give an output standard deviation of near zero cec over the operating range of SNR, this value was initially chosen. However, it was found that this amount of first-order filtering reduced the data rate to below acceptable limits (i.e. in step #2 the lock-up time was found to be prohibitive). Therefore, the first-order filter constant of 4 bits (16 counts or .05 Hz bandwidth) was chosen.

The .05 Hz first-order loop bandwidth yields acceptable lock-up times for several values of second-order gain (again, determined by executing step #2). Therefore, the best value of second-order gain is chosen as that which provides best steady-state performance for ramp inputs over the desired range of input SNR's (step #3). Figure E-9 shows the steady-state response of SMAPLL2 to a zero-velocity input for several values of second-order gain. Figure E-10 gives the same data for a 300-knot input ramp. It can be seen that a value of .25 for the second-order gain gives the best response for both input velocities.

ORIGINAL PAGE IS
OF POOR QUALITY

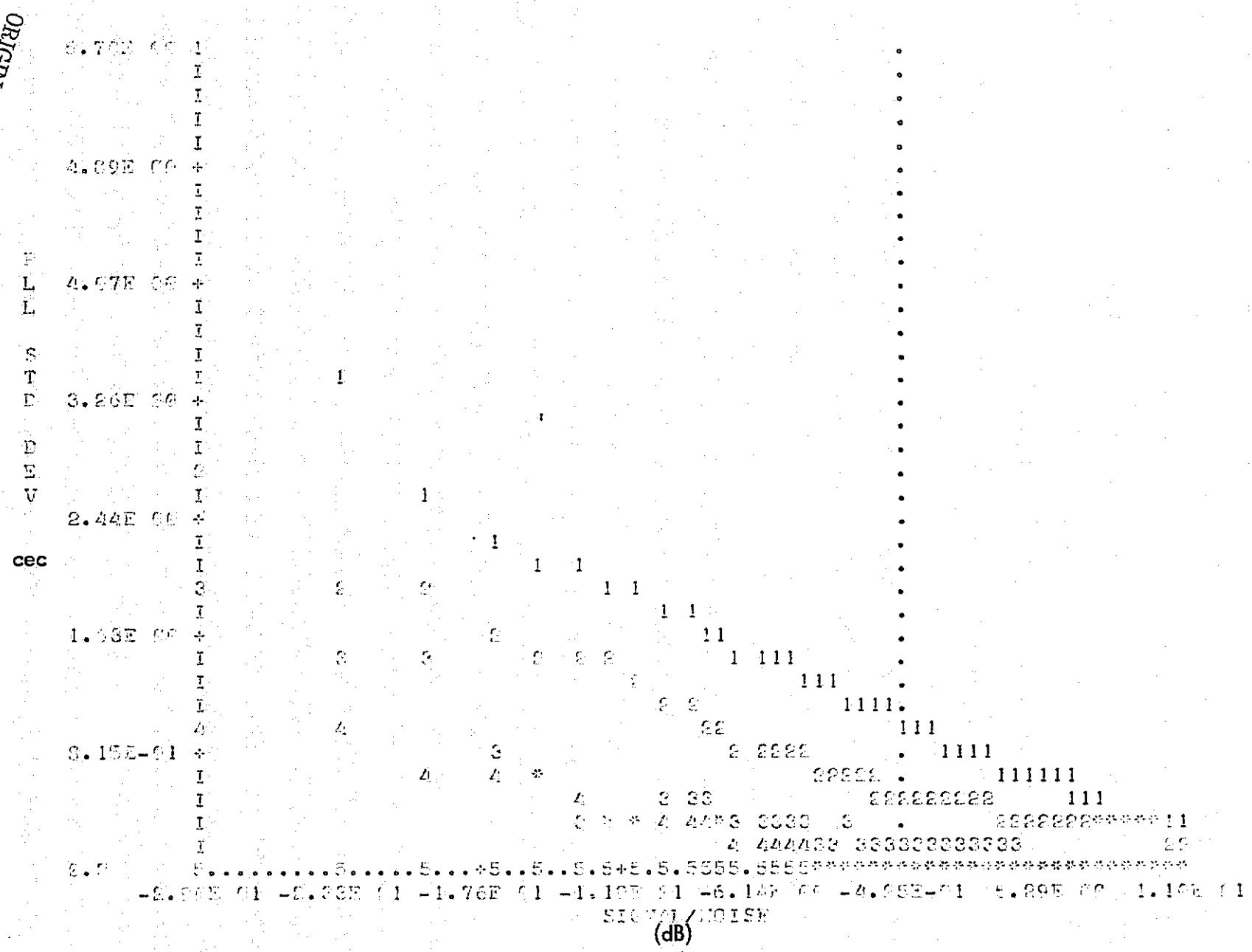


Figure E-8. SMAPLL1 Steady-State Response to O-Velocity Input for First-Order Filter Equal to 2 Counts (1), 4 counts (2), 8 counts (3), 16 counts (4) and 32 counts (5).

ORIGINAL PAGE IS
OF POOR QUALITY

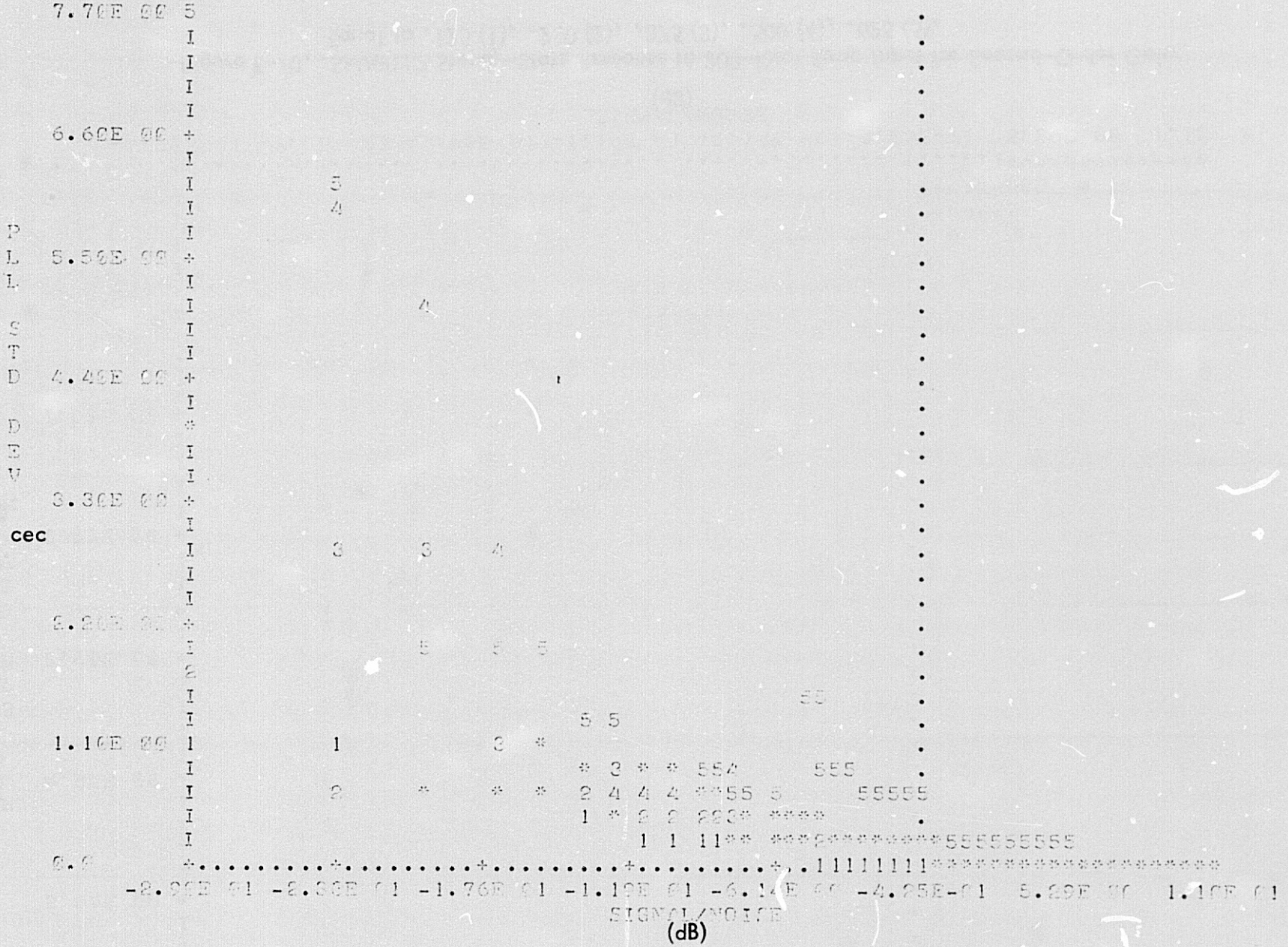


Figure E-9. SMAPLL2 Steady-State Response to Zero-Velocity Input for Second-Order Gain Equal to Zero (1), .125 (2), .250 (3), .375 (4), .500 (5).

ORIGINAL PAGE IS
OF POOR QUALITY

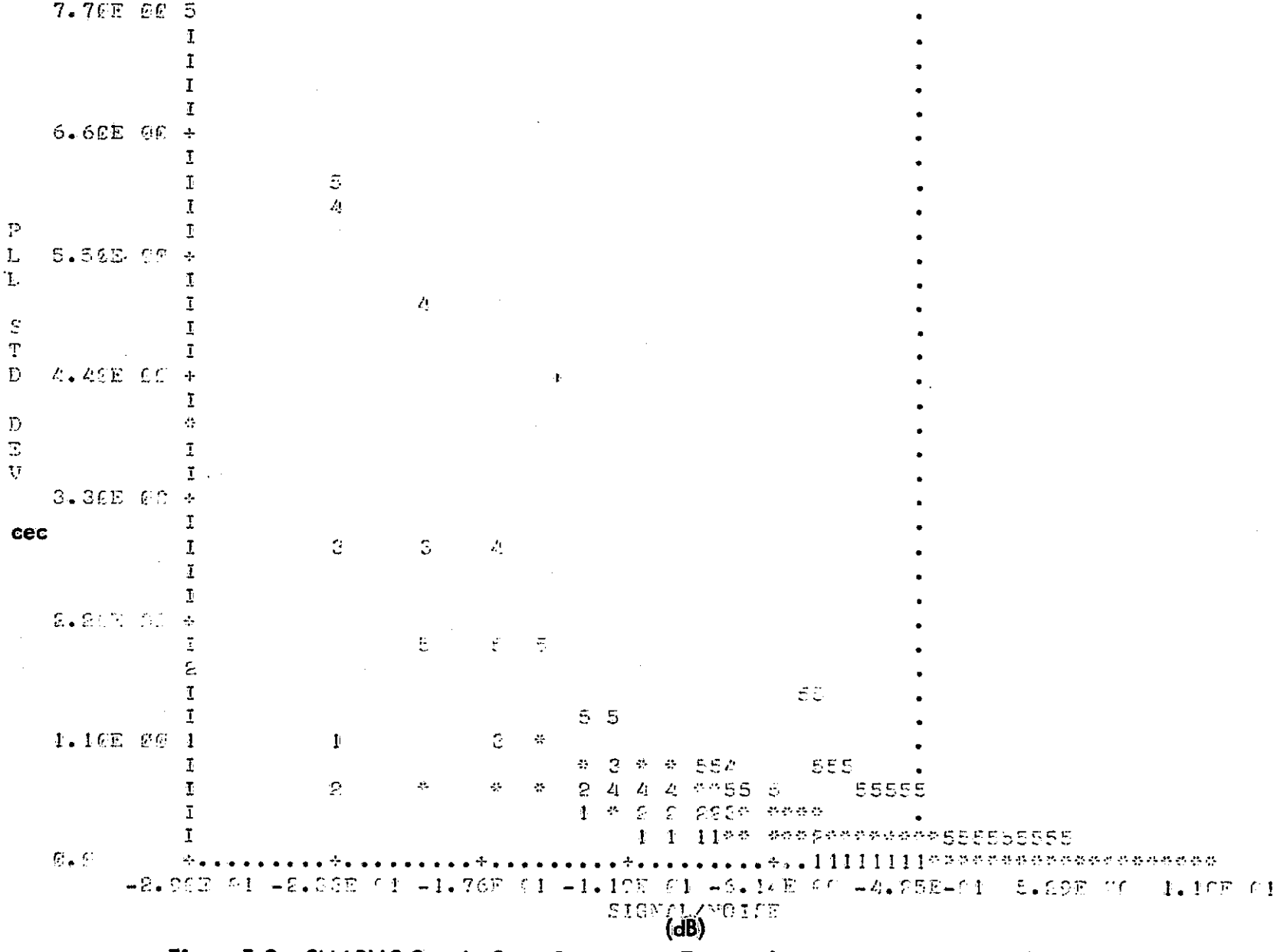


Figure E-9. SMAPLL2 Steady-State Response to Zero-Velocity Input for Second-Order Gain Equal to Zero (1), .125 (2), .250 (3), .375 (4), .500 (5).

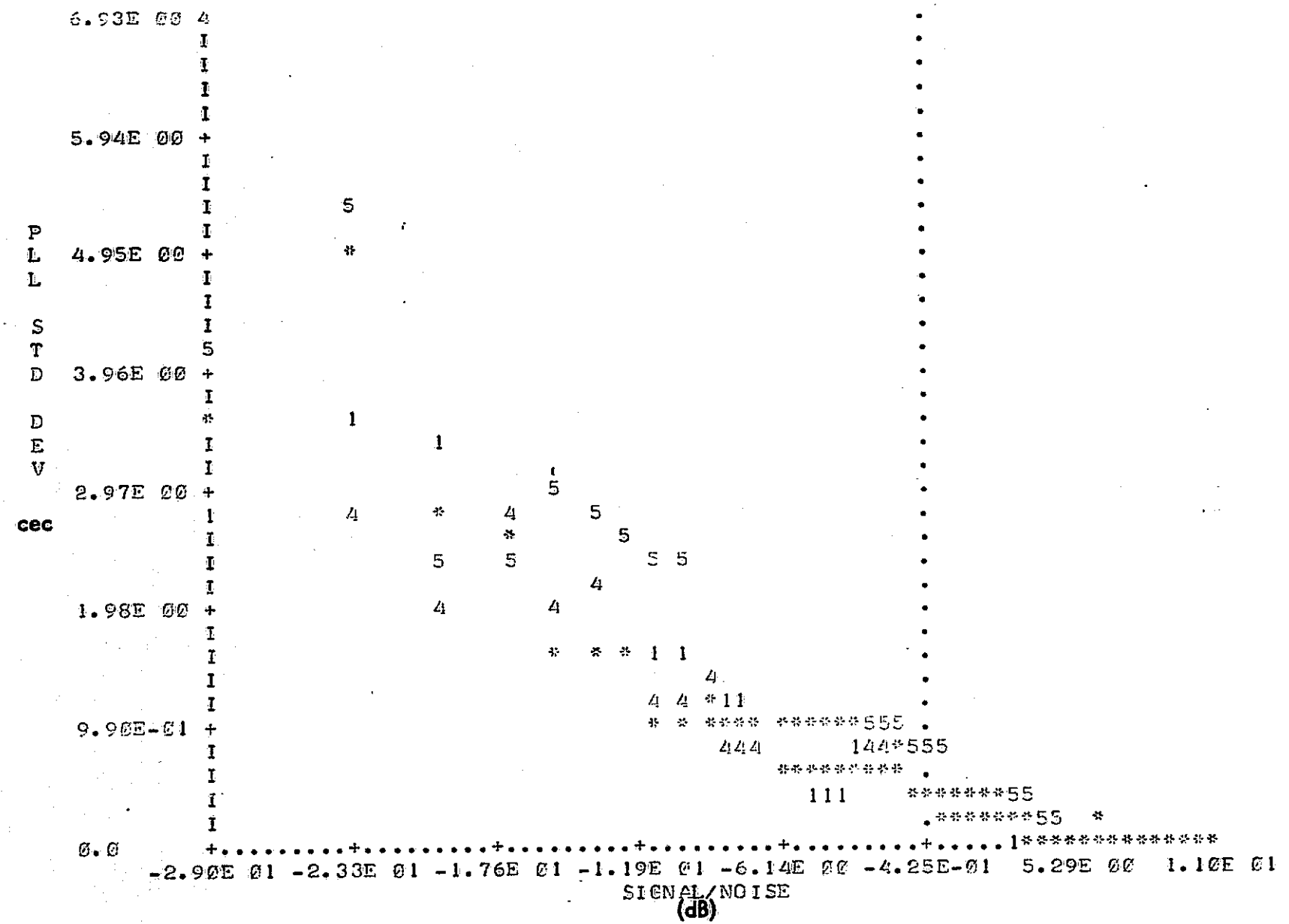


Figure E-10. SMAPLL2 Steady-State Response to 300-Knot Ramp Input for Second-Order Gain Equal to .125 (1), .250 (2), .375 (3), .500 (4), .625 (5).

The transient response of the optimized SMAPLL2 to 180 degree step and 300-knot ramp inputs is plotted in Figures E-11 and E-12. Since a general aviation aircraft is usually on the ground for at least ten minutes after aircraft power is turned on, a lock-up time of this length is considered acceptable.

The FORTRAN language listings for computer simulations of Omega noise, the automatic synchronization algorithm, and the performance simulation of the SMAPLL2 are given as Figures E-13, E-14 and E-15, respectively.

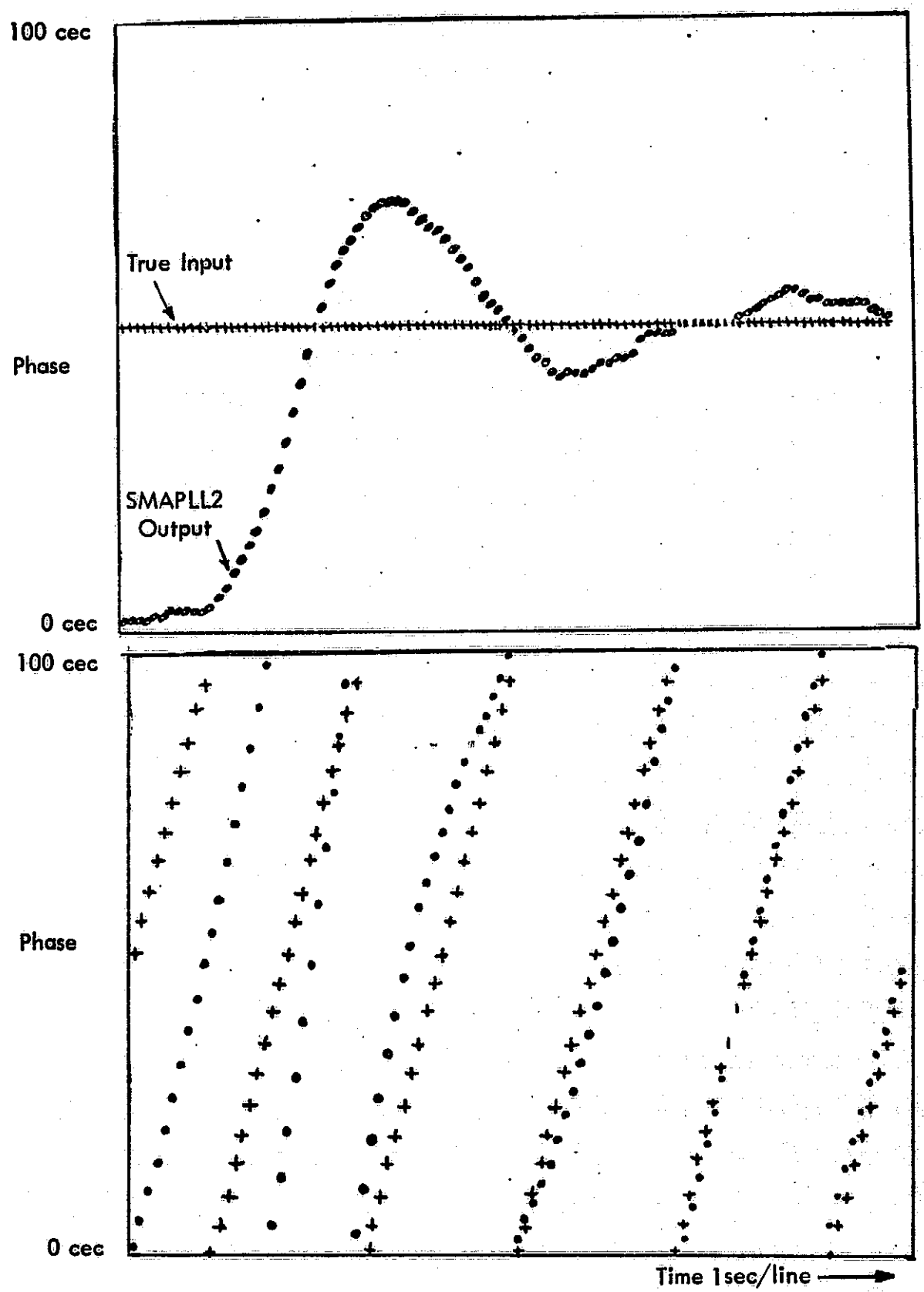


Figure E-11. SMAPLL2 Transient (Lock-Up Mode) Response to 180° Step (Top) and 300-Knot Ramp (Bottom) Input of -10 dB SNR.

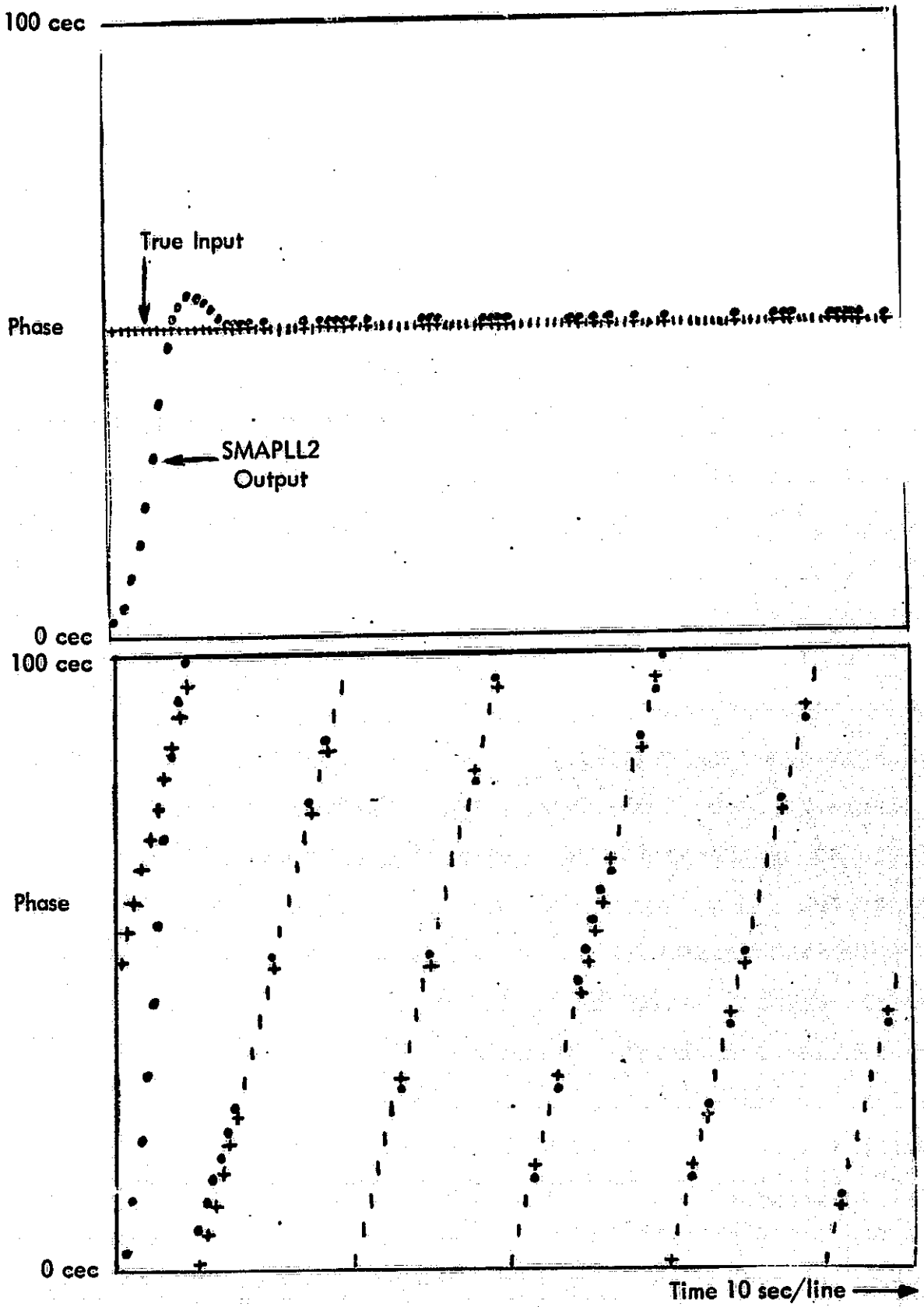


Figure E-12. SMAPLL2 Transient (Lock-Up Mode) Response to 180° Step(Top) and 300-Knot Ramp (Bottom) Inputs at +10 dB SNR.

```
C     ROUTINE COMPUTES THE AUTOCORRELATION R(T) OF BOTH REAL OMEGA
C     NOISE AND SIMULATED NOISE, AND PLOTS R(T) ON SEPARATE GRAPHS.
C
C     ROUTINE COMPUTES AND PLOTS THE HISTOGRAMS OF BOTH REAL OMEGA
C     NOISE AND SIMULATED NOISE PHASE ON SEPARATE GRAPHS.
C
C     OPERATOR MUST SELECT SNR FOR SIMULATED NOISE BY SETTING A.
C     OPERATOR MUST SELECT SNR FOR REAL OMEGA NOISE BY READING OVER
C     DATA ON DATA BASE WHICH IS NOT WANTED.
C
      INTEGER TME
      REAL X(200),XX(100),R(100),XXA(100)
      REAL Y(200),YY(200),XXL(12),YYL(12),INTVL
      DATA XXL/'T','I','M','E',' ','1','0',' ','M','S','E','C'/
      DATA YYL/'A','U','T','O','C','O','R','R','E','L','A','T'/
C     CALL RANDOM NUMBER GENERATOR AND LPF TO GET SIMULATED NOISE.
C     SET PARAMETERS ON LPF TO SIMULATE 30HZ BW AND 100HZ SAMPLES.
      CALL INTGEN(5**11)
      BW=15.
      SMALLA=BW*12.57
      CAPA=2.*SQRT(4./SMALLA)
      INTVL=.01
      TME=200
      CALL GRANGE(YY,INTVL,CAPA,SMALLA,TME)
      CALL INTGEN(5**17)
      CALL GRANGE(Y,INTVL,CAPA,SMALLA,TME)
C     TAKE ARCTAN OF RANDOM NUMBER RATIOS TO GET NOISE PHASE.
C     A IS THE SIGNAL VOLTAGE AMPLITUDE: POWER SNR IS (A*A)/(2*N*N).
  555 DO 5 I=1,200
      A=0.001
      X(I)=ATAN2(YY(I),(Y(I)+A))
C     CHANGE UNITS FROM RADIANS TO CEC.
      X(I)=X(I)/0.062831853
    5 CONTINUE
C     CALL AUTCOR SUBROUTINE TO COMPUTE AND PLOT R(T).
      CALL AUTCOR(X)
      PRINT 102
  102 FORMAT(1X,'THIS IS THE AUTOCORRELATION FOR SIMULATED NOISE PHASE')
      SNR=10.*ALOG10(A*A*.5)
      BW2=2.*BW
      PRINT 101,SNR,BW2,INTVL
  101 FORMAT(1X,10X,'SNR = ',F6.2,' DB',10X,'BW = ',F3.0,' HZ',10X,
     C'SAMPLING TIME = ',F5.3,' SEC')
C     CALL TO HISTO SUBROUTINE PLOTS HISTOGRAM OF SIMULATED PHASE
C     NOISE VS. PHASE IN CEC.
      CALL HISTO(X)
      PRINT 105
  105 FORMAT(1X,'THIS IS THE HISTOGRAM OF SIMULATED NOISE PHASE  VS.
     CPHASE IN CEC.')
      PRINT 101,SNR,BW2,INTVL
C
C     CALL DATA SUBROUTINE TO RETURN REAL OMEGA PHASE PLUS NOISE.
C     FIRST SELECT SNR OF REAL DATA BY READING OVER UNWANTED DATA.
  500 CONTINUE
      DO 1 I=1,130
```

Figure E-13. FORTRAN Program RSRAND Used to Analyze Real and Simulated Omega Noise.

ORIGINAL PAGE IS OF POOR QUALITY

```
      CALL DATA(IOUT,&100)
      ISKIP=I
    1 CONTINUE
      DO 2 I=1,100
      CALL DATA(IOUT,&100)
      X(I)=FLOAT(IOUT)
    2 CONTINUE
      LANE=255
      HALFLN=128.
      SUM=0
      DO 6 I=1,100
      SUM=SUM+X(I)
    6 CONTINUE
      AVE=SUM/100.
      DO 7 I=1,100
      X(I)=X(I)-AVE
      IF(X(I).GT.HALFLN)X(I)=X(I)-LANE
      IF(X(I).LT.-HALFLN)X(I)=X(I)+LANE
C     CHANGE UNITS FROM 255THS TO CEC.
      X(I)=X(I)*100./255.
    7 CONTINUE
      DO 3 I=1,900
      CALL DATA(IOUT,&100)
    3 CONTINUE
      DO 4 I=101,200
      CALL DATA(IOUT,&100)
      X(I)=FLOAT(IOUT)
    4 CONTINUE
      SUM=0.
      DO 8 I=1,100
      SUM=SUM+X(100+I)
    8 CONTINUE
      AVE=SUM/100.
      DO 9 I=101,200
      X(I)=X(I)-AVE
      IF(X(I).GT.HALFLN)X(I)=X(I)-LANE
      IF(X(I).LT.-HALFLN)X(I)=X(I)+LANE
C     CHANGE UNITS FROM 255THS TO CEC:
      X(I)=X(I)*100./255.
    9 CONTINUE
C     CALL AUTCOR SUBROUTINE TO COMPUTE AND PLOT R(T).
      CALL AUTCOR(X)
      PRINT 103
  103 FORMAT(1X,'THIS IS THE AUTOCORRELATION OF REAL OMEGA NOISE')
      PRINT 104,ISKIP
  104 FORMAT(1X,'THE DATA STARTED ',I5,'SAMPLES AFTER THE START OF
     CTHE D TIME SLOT')
C     CALL THE HISTOGRAM SUBROUTINE FOR REAL OMEGA NOISE PHASE .
      CALL HISTO(X)
      PRINT 106
  106 FORMAT(1X,'THIS IS THE HISTOGRAM OF REAL OMEGA NOISE PHASE VS.
     C PHASE IN CEC.')
      PRINT 104,ISKIP
  100 STOP
      END
```

ORIGINAL PAGE IS
OF POOR QUALITY

Figure E-13. (Continued)

```
C     XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
      SUBROUTINE AUTCOR(X)
      REAL X(200),XX(100),R(100),XXA(100),XXL(12),YYL(12)
      DATA XXL/'T','I','M','E',' ','1','0',' ','M','S','E','C'/
      DATA YYL/'A','U','T','O','C','O','R','R','E','L','A','T'/
C     COMPUTE AND PLOT AUTOCORRELATION OF THE NOISE.
      DO 10 I=1,100
      XX(I)=X(I)
   10 CONTINUE
C     COMPUTE  AUTOCORRELATION FOR ONE VALUE OF TAU.
      DO 20 II=1,100
      SUM=0.
      DO 30 I=1,100
      IF(II.EQ.1)GO TO 16
      GO TO 15
   16 J=I
      GO TO 17
   15 J=I+II
   17 CONTINUE
      PROD=XX(I)*X(J)
      SUM=SUM+PROD
   30 CONTINUE
      R(II)=SUM/100.
   20 CONTINUE
      DO 50 I=1,100
      XXA(I)=FLOAT(I)
   50 CONTINUE
      CALL MMPLOT(XXA,R,100,1,XXL,YYL)
      RETURN
      END
C     XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
      SUBROUTINE GRANGE(Y,INTVL,A,SMALLA, TME)
      INTEGER TME
      REAL Y(5000), INTVL
      TAU=INTVL
      R=EXP(-SMALLA*TAU)
      PSI=(A*A*SMALLA)/4.
      CONS=SQRT(PSI*(1.-(R*R)))
      CALL ANORM(AN)
      Y(1)=AN*PSI
      DO 10 I=2,TME
      J=I-1
      CALL ANORM(AN)
    9 FORMAT(1X,10E10.2)
      Y(I)=AN*CONS+R*Y(J)
   10 CONTINUE
      RETURN
      END
C     XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
      SUBROUTINE DATA(IOUT,*)
C...DEBLOCKS OMEGA TAPE DATA AND RETURNS ONE DATA SAMPLE TO MAIN PROGRA
C...
C...ON TAPE EOF, RETURNS TO *
C...
C... R. W. LILLEY, AVIONICS, JANUARY, 1976
```

Figure E-13. (Continued)

```
      INTEGER I/0/
      LOGICAL*1 IN(1000),INB(4)
      EQUIVALENCE(II,INB(1))
      II=0
      IF(I.EQ.0)GO TO 10
   30 INB(4)=IN(I)
      IOUT=II
      I=I+1
      IF(I.GT.1000)I=0
      RETURN
C...
   10 I=1
   17 READ(11,5,END=20)IN
    5 FORMAT(5(200A1))
      GO TO 30
C...
   20 RETURN 1
C...
      ENTRY START
C...STARTS DATA OUTPUT AT A RANDOM POINT ON TAPE. READS UP TO
C...  25 RECORDS, AND STARTS BETWEEN BYTE 1-1000.
C...
C...SEED WITH UNDEFINED VARIABLE
      IF(IQ.EQ.0)IQ=5**13
      IQ=IABS(IQ/2)*2+1
      CALL INTGEN(IQ)
      RETURN
      ENTRY RAND(NBYTE)
      CALL IRAND(NREC,24)
      CALL IRAND(NBYTE,999)
C...
      DO 51 IR=1,NREC
   51 READ(11,5)IN
      I=NBYTE+1
      PRINT 52,NREC,NBYTE
   52 FORMAT(' THIS RUN STARTS AFTER '/1X,I2,' SKIPPED RECORDS AND',
     *1X,I4,' SKIPPED BYTES ')
      GO TO 17
      END
C     XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
      SUBROUTINE ILPF(Y,TAU,A,BW,NUM)
      PRINT 14,TAU,BW
   14 FORMAT(1X,2F10.3)
C     ROUTINE RETURNS 'NUM' IDEAL LOW-PASS-FILTERED, GAUSSIANLY
C     DISTRIBUTED RANDOM VARIABLES.
C
C     Y IS THE VECTOR OF SIN(X)/X CORRELATED NUMBERS;TAU IS THE INDEP-
C     ENDENT TIME SHIFT VARIABLE; A IS A WASTE; BW IS THE BANDWIDTH
C     OF THE FILTER(ONE-SIDED); NUM IS THE NUMBER OF NUMBERS RETURNED.
      REAL Y(5000)
      R=(SIN(6.283185*BW*TAU))/(6.283185*BW*TAU)
      CONS=SQRT(ABS(1.-(R*R)))
      PRINT 12,R,CONS
   12 FORMAT(1X,2F10.3)
      CALL ANORM(AN)
```

Figure E-13. (Continued)

```
      Y(1)=AN
      DO 10 I=2,NUM
      J=I-1
      CALL ANORM(AN)
      Y(I)=AN*CONS+R*Y(J)
   11 FORMAT(1X,2F8.2)
   10 CONTINUE
      RETURN
      END
C     XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
      SUBROUTINE HISTO(X)
C     ROUTINE PLOTS HISTOGRAM OF NUMBERS IN VECTOR (X(200)).
      INTEGER IX(200)
      REAL X(200), XA(101), YA(101), XL(12), YL(12)
      DATA XL/'N','O','I','S','E',' ','P','H','A','S','E',' '/
      DATA YL/'H','I','S','T','O','G','R','A','M',' ',' ',' '/
      DO 10 I=1,200
      IX(I)=INT(X(I))
   10 CONTINUE
C     CREATE X-AXIS VECTOR OF -50. TO 50. CEC.
      DEV=-50.
      DO 60 I=1,101
      YA(I)=0.
      XA(I)=DEV
      DEV=DEV+1.
   60 CONTINUE
C     CREATE Y-AXIS VECTOR OF FREQ. OF OCCURENCE OF EACH CEC VALUE.
      DO 70 I=1,200
      DO 80 J=1,101
      IF(IX(I).EQ.INT(XA(J)))YA(J)=YA(J)+1.
   80 CONTINUE
   70 CONTINUE
C     NORMALIZE RELATIVE FREQ. OF OCCURENCE.
      DO 90 I=1,101
      YA(I)=YA(I)/200.
   90 CONTINUE
      CALL MMPLOT(XA,YA,101,1,XL,YL)
      RETURN
      END
```

Figure E-13. (Continued)

```
C       OMEGA SENSOR PROCESSOR SIMULATION.
C       MAIN PROGRAM
C
C       R. SALTER,AVIONICS ENGINEERING CENTER, MARCH 1976
C
C       ************************************************************
C       OMEGA AUTO-SYNC ROUTINE
        INTEGER LOP(1000)
        INTEGER X(1000),DIFF(1000),C,LIVSLT(1000),CORR,CTR,SYNCPT
        INTEGER SLOT(1000)
        INTEGER SMDIFF(1000)
        INTEGER STA(8)
        INTEGER WITHIN,V(1000)
        INTEGER INMIR(1000)
        INTEGER PHASE(8)
        INTEGER XX(2048),YY(2048),XL(12),YL(12)
        INTEGER*2 TSLOT(8)/90,100,110,120,90,110,120,100/
        INTEGER LCW(8),PHI
        INTEGER CPHI(8,1000),CCPHI(1000)
        INTEGER II(8),SOGAIN(8),VEL(8),RATE
        INTEGER CONST,SUMDIF(8),VAR(8,1000),VVAR(1000)
C       DERIVE RANDOM STARTING POINT ON DATA-BASE TAPE
C       SKIP 'NSKIP' FILES ON DATA BASE TAPE.
        NSKIP=4
        CALL START
        CALL RAND(NBYTE)
C       TAKE ONE RECORD (1000 BYTES) OF DATA AND MIRROR IT
        DO 1 I=1,1000
        CALL DATA(X(I),&50)
        CALL MIRR(X(I),INMIR(I))
      1 CONTINUE
        CALL PLOT10(X,1)
        CALL PLOT10(INMIR,1)
C       OBTAIN A 1000 BYTE VECTOR OF DIFFERENCES TO BE USED AS A MEASURE
C       OF PHASE COHERENCE IN EACH 10 MSEC SAMPLE SLOT
        CALL DIFFER(INMIR,DIFF)
        CALL PLOT10(DIFF,1)
      4 FORMAT(1X,25I4)
C       SMOOTH DIFFERENCES (OPTIONAL).
        CALL SMOOTH(DIFF,SMDIFF)
        PRINT 151
    151 FORMAT(1X,'9 PT. STRAIGHT AVERAGING SMOOTHER INSTALLED')
C       GENERATE A 1000 BYTE VECTOR OF 1'S AND 0'S REPRESENTING  SIGNAL
C       AND NOISE SLOTS
C       MUST SET THRESHOLD "C" TO COMPARE DIFFERENCES TO.
        C=120
     12 CONTINUE
        CALL COMPAR(C,SMDIFF,LIVSLT)
C       PERFORM BILEVEL CORRELATION OF KNOWN OMEGA PATTERN TO LIVE DATA
        CORR=0
        J=0
        CALL PATERN(SLOT)
     10 CALL BICOR(LIVSLT,CTR,SLOT)
        I=J+1
        IF(CTR.GE.CORR) CORR=CTR
```

ORIGINAL PAGE IS
OF POOR QUALITY

Figure E-14. FORTRAN Program RSOSP Used to Simulate Automatic Synchronization Routine.

```
      IF(CTR.EQ.CORR)SYNCPT=J
      J=J+1
      IF(J.GT.1000)GO TO 100
      NSHIFT=1
      CALL SHIFT(LIVSLT,NSHIFT)
      GO TO 10
C     PRINT CORRELATION VALUE AND SYNCPOINT VALUE OBTAINED.
  100 PRINT 110,C,CORR,SYNCPT
  110 FORMAT(1X,'C = ',I3,3X,' CORR = ',I4,3X,'SYNCPT = ',I4)
      WITHIN=10*(380+NBYTE-SYNCPT)
      PRINT 13,WITHIN
   13 FORMAT(1X,'SYNC WITHIN',I5,1X,'MSEC')
      NSHIFT=SYNCPT
C     SHIFT VECTOR OF PHASE VALUES TO SYNCPOINT AND PLOT THEM OUT.
      CALL SHIFT(X,NSHIFT)
      CALL PLOT10(X,1)
      GO TO 50
      PRINT 111
  111 FORMAT(1X,'ALL STATIONS ON AIR')
C     RESET DATA BASE TAPE TO SYNCPOINT.
      NSHIFT=1000-NSHIFT
      DO 14 I=1,NSHIFT
      CALL DATA (IOUT,850)
   14 CONTINUE
C     ********************************************************************
C     OMEGA BURST FILTER ROUTINE.
C     PERFORM BURST PLL FILTERING TO INCOMING OMEGA PHASE MEASURFMENTS.
      DO 18 I=1,8
      LCW(I)=0
      SOGAIN(I)=0
      VEL(I)=0
   18 II(I)=1
C     READ 1000 OMEGA FRAMES.
      DO 26 K=1,1000
C     READ 8 TIME SLOTS IN EACH OMEGA FRAME.
   19 DO 25 I=1,8
      JJ=TSLOT(I)
C     READ TSLOT(I) PHASE MEASUREMENTS IN EACH TIME SLOT.
      LPCTR=0
      RATE=0
      SUMDIF(I)=0
      DO 20 J=1,JJ
      CALL DATA(PHI,850)
      CONST=2
      CALL SMAPLL(PHI,LPCTR,LCW(I),CONST,RATE,SUMDIF(I))
      IDIF=SUMDIF(I)
      LLCW=LCW(I)
   21 FORMAT(1X,5I10)
   20 CONTINUE
   22 FORMAT(1X)
      CPHI(I,II(I))=LCW(I)
C     COMPUTE SIGNAL VARIANCE IN TIMESLOT.
      SUMDIF(I)=SUMDIF(I)/TSLOT(I)
      VAR(I,II(I))=SUMDIF(I)
      IF(RATE.LT.0)VEL(I)=VEL(I)-RATE*SOGAIN(I)
```

Figure E-14. (Continued)

```
      IF(RATE.GT.0)VEL(I)=VEL(I)+RATE*SOGAIN(I)
      LCW(I)=LCW(I)+VEL(I)
      IF(LCW(I).LT.0)LCW(I)=256+LCW(I)
      IF(LCW(I).GT.255)LCW(I)=LCW(I)-256
      II(I)=II(I)+1
C     SKIP SAMPLES FROM .2 SEC. NOISE GAP.
      DO 23 K=1,20
   23 CALL DATA(PHI,850)
   25 CONTINUE
   26 CONTINUE
C     DUMP EACH STATION'S TIMESLOT PHASE ESTIMATE INTO VECTOR TO BE PLO-
C     TTED BY PLOT1 SUBROUTINE.
   30 DO 40 I=1,8
      DO 35 J=1,1000
      CCPHI(J)=CPHI(I,J)
C     DUMP EACH STATIONS VARIANCE INTO A VECTOR TO BE PRINTED OUT.
      VVAR(J)=VAR(I,J)
   35 CONTINUE
C     ********************************************************************
C     OMEGA LOP ROUTINE
      PRINT 36,I
   36 FORMAT('   PHASE OF STATION ',I2,' MINUS CLOCK')
      PRINT 60,CCPHI
   60 FORMAT(1X,10I10)
      PRINT 61,I
   61 FORMAT('    SIGNAL VARIANCE OF STATION ',I2)
      PRINT 60 , VVAR
      CALL PLOT1(CCPHI,1)
   40 CONTINUE
      ISTA1=3
      ISTA2=4
      CALL LOPO(CPHI,ISTA1,ISTA2,LOP)
C     PLOT LOP SPECIFIED BY ISTA1-ISTA2.
      CALL PLOT1(LOP,1)
      STOP
   50 STOP 50
      END
      SUBROUTINE DIFFER(X,DIFF)
C     OMEGA AUTO-SYNC SIMULATION
C
C     ROUTINE TAKES 1000BYTE VECTOR OF INPUT PHASE DATA AND RETURNS A
C     DIFFERENCE VALUE WHICH IS INVERSELY PROPORTIONAL TO THE PHASE
C     COHERENCE (SIGNAL STRENGTH) IN EACH 10 MSEC SAMPLE SLOT.
C
C     R. SALTER, AVIONICS ENGINEERING CENTER, MARCH 1976
C
      INTEGER DIFF(1000),X(1000)
      DO 20 I=1,999
      J=I+1
      DIFF(I)=IABS(X(I)-X(J))
   20 CONTINUE
      DIFF(1000)=IABS(X(1)-X(1000))
      RETURN
      END
      SUBROUTINE COMPAR(C,DIFF,LIVSLT)
```

ORIGINAL PAGE IS
OF POOR QUALITY

Figure E-14. (Continued)

```
C       OMEGA AUTO-SYNC SIMULATION
C
C       ROUTINE COMPARES 1000 BYTE VECTOR OF DIFFERENCES TO THRESHOLD
C       CONSTANT AND RETURNS A BILEVEL VECTOR CONTAINING 1'S FOR SIGNAL
C       AND 0'S FOR NOISE
C
C       R. SALTER, AVIONICS ENGINEERING CENTER, MARCH 1976
C
        INTEGER C,DIFF(1000),LIVSLT(1000)
        DO 10 I=1,1000
        IF(DIFF(I).GE.C)LIVSLT(I)=0
        IF(DIFF(I).LT.C)LIVSLT(I)=1
   10   CONTINUE
        RETURN
        END
        SUBROUTINE BICOR(LIVSLT,CTR,SLOT)
C       OMEGA AUTO-SYNC SIMULATION
C
C       ROUTINE PERFORMS BILEVEL CROSS-CORRELATION OF OMEGA TRANSMISSION
C       PATTERN OFF AIR WITH THE KNOWN PATTERN GENERATED BY "PATTERN" ,
C       AND RETURNS VALUE PROPORTIONAL TO CORRELATION
C
C       R. SALTER, AVIONICS ENGINEERING CENTER, MARCH 1976
C
        INTEGER LIVSLT(1000),CTR
        INTEGER SLOT(1000)
        CTR=0
        DO 10 I=1,1000
        IF(LIVSLT(I).EQ.SLOT(I))CTR=CTR+1
   10   CONTINUE
        RETURN
        END
        SUBROUTINE SHIFT(LIVSLT,NSHIFT)
C       OMEGA AUTO-SYNC SIMULATION
C
C       ROUTINE SHIFTS LIVSLT VECTOR "NSHIFT" SLOTS TO THE RIGHT.
C
C       R. SALTER, AVIONICS ENGINEERING CENTER, MARCH 1976
C       MODIFIED BY R. SALTER, APRIL,1976
C
        INTEGER LIVSLT(1000),AUXSLT(1000)
        DO 30 K=1,NSHIFT
        DO 10 I=2,1000
        J=I-1
        AUXSLT(I)=LIVSLT(J)
   10   CONTINUE
        AUXSLT(1)=LIVSLT(1000)
        DO 20 I=1,1000
        LIVSLT(I)=AUXSLT(I)
   20   CONTINUE
   30   CONTINUE
        RETURN
        END
        SUBROUTINE PATERN(SLOT)
C       OMEGA AUTO-SYNC SIMULATION
```

Figure E-14. (Continued)

```
C
C     ROUTINE CREATES KNOWN OMEGA TRANSMISSION PATTERN
C
C     R. SALTER, AVIONICS ENGINEERING CENTER, MARCH 1976
C
      INTEGER SLOT(1000),STA(8)
      INTEGER JJ(8)/1,111,231,361,501,631,741,881/
      INTEGER KK(8)/90,210,340,480,610,720,860,980/
      READ(1,1)STA
    1 FORMAT(8I1)
      DO 10 I=1,8
      J=JJ(I)
      K=KK(I)
  100 DO 110 L=J,K
      SLOT(L)=STA(I)
  110 CONTINUE
      M=K+1
      N=K+20
      DO 120 L=M,N
      SLOT(L)=0
  120 CONTINUE
   10 CONTINUE
  130 RETURN
      END
      SUBROUTINE DATA(IOUT,*)
C...DEBLOCKS OMEGA TAPE DATA AND RETURNS ONE DATA SAMPLE TO MAIN PROGRAM
C...
C...ON TAPE EOF, RETURNS TO *
C...
C...  R. W. LILLEY, AVIONICS, JANUARY, 1976
      INTEGER I/0/
      LOGICAL*1 IN(1000),INB(4)
      EQUIVALENCE(II,INB(1))
      II=0
      IF(I.EQ.0)GO TO 10
   30 INB(4)=IN(I)
      IOUT=II
      I=I+1
      IF(I.GT.1000)I=0
      RETURN
C...
   10 I=1
   17 READ(11,5,END=20)IN
    5 FORMAT(5(200A1))
      GO TO 30
C...
   20 RETURN 1
C...
      ENTRY START
C...STARTS DATA OUTPUT AT A RANDOM POINT ON TAPE. READS UP TO
C...  25 RECORDS, AND STARTS BETWEEN BYTE 1-1000.
C...
C...SEED WITH UNDEFINED VARIABLE
      IF(IQ.EQ.0)IQ=5**13
      IQ=IABS(IQ/2)*2+1
```

Figure E-14. (Continued)

```
      CALL INTGEN(IQ)
      RETURN
      ENTRY RAND(NBYTE)
      CALL IRAND(NREC,24)
      CALL IRAND(NBYTE,999)
C...
      DO 51 IR=1,NREC
   51 READ(11,5)IN
      I=NBYTE+1
      PRINT 52,NREC,NBYTE
   52 FORMAT(' THIS RUN STARTS AFTER '/1X,I2,' SKIPPED RECORDS AND',
     *1X,I4,' SKIPPED BYTES ')
      GO TO 17
      END
      SUBROUTINE MIRR(IN,INMIR)
C...OMEGA DATA SIMULATION
C...ROUTINE REFLECTS COUNTS 128-255 MIRRORED TO 127-0
C...
C... R. W. LILLEY, AVIONICS, JANUARY, 1976
C     MODIFIED BY P. SALTER, APRIL,1976
      IF(IN.LE.127)GO TO 10
      INMIR=255-IN
      RETURN
   10 INMIR=IN
      RETURN
      END
      SUBROUTINE SMOOTH(DIFF,SMDIFF)
C     OMEGA AUTO-SYNC SIMULATION
C
C     ROUTINE SMOOTHS VECTOR OF 1000 DIFFERENCES BY PASSING EACH SAMPLE
C     POINT THRU A 9 POINT EXPONENTIAL FILTER
C
C     P. SALTER, AVIONICS ENGINEERING CENTER, MARCH 1976
C
      INTEGER SMDIFF(1000)
      INTEGER DIFF(1000)
      DO 10 I=1,1000
      IF(I.LT.5)GO TO 20
      IF(I.GT.995)GO TO 30
      J=I-1
      K=I-2
      L=I-3
      M=I-4
   25 JJ=I+1
      KK=I+1
      LL=I+3
      MM=I+4
C     TEMPORARY SMOOTHER --- NINE PT. STRAIGHT AVERAGER.
C     USES 4 PTS. BEHIND , 4 PTS. AHEAD, AND CURRENT PT. EQUALLY WEIGHT
   35 SMDIFF(I)=DIFF(I)+DIFF(J)+DIFF(K)+DIFF(L)+DIFF(M)+DIFF(JJ)+DIFF
     *(KK)+DIFF(LL)+DIFF(MM)
   10 CONTINUE
      GO TO 40
   20 J=1000-MOD(1000-I+1,1000)
      K=1000-MOD(1000-I+2,1000)
```

Figure E-14. (Continued)

```
      L=1000-MOD(1000-I+3,1000)
      M=1000-MOD(1000-I+4,1000)
      GO TO 25
   30 J=I-1
      K=I-2
      L=I-3
      M=I-4
      JJ=MOD(I+1,1000)
      KK=MOD(I+2,1000)
      LL=MOD(I+3,1000)
      MM=MOD(I+4,1000)
      GO TO 35
   40 RETURN
      END
      SUBROUTINE PLOT10(X,LLL)
C     ROUTINE PLOTS  DATA VECTOR "X" 10 POINTS PER LINE WITH STARS...
C
C     SCALE OF PLOT IS 1-128... IF VALUES OF "X" ARE LESS THAN 128 THEN
C     OPERATOR SHOULD SET LLL=0 AND A STRAIGHT PLOT WILL RESULT. IF "X"
C     HAS VALUES BETWEEN 128-256 LLL SHOULD BE SET TO 1 AND ROUTINE WILL
C     HALVE EACH VALUE BEFORE PLOTTING.
C
C     R. SALTER, AVIONICS ENGINEERING CENTER, APRIL,1976.
C
      INTEGER BK/' '/,ST/'*'/,LIN(128),X(1000)
      K=0
      DO 30 J=1,991
      DO 10 I=1,128
   10 LIN(I)=BK
      J=K+1
      K=J+9
      DO 20 I=J,K
      IF(LLL.EQ.0)GO TO 12
      MM=X(I)/2+1
      GO TO 13
   12 CONTINUE
      MM=X(I)
   13 CONTINUE
      LIN(MM)=ST
   20 CONTINUE
      PRINT 11,LIN
   11 FORMAT(1X,128A1)
   30 CONTINUE
      RETURN
      END
      SUBROUTINE LOPO(CPHI,ISTA1,ISTA2,LOP)
C     ROUTINECREATES A VECTOR OF LOP POINTS (LOP(1000)) BY DIFFERENCING
C      STA1-STA2 PHASE MEASUREMENTS SUPPLIED IN CPHI(8,1000) ARRAY.
C     R. SALTER, AVIONICS ENGINEERING CENTER, OHIO UNIVERSITY, JULY 1976
C
      INTEGER LOP(1000),CPHI(8,1000)
      DO 10 I=1,1000
      LOP(I)=CPHI(ISTA1,I)-CPHI(ISTA2,I)
      IF(LOP(I).LT.0)LOP(I)=LOP(I)+256
   10 CONTINUE
```

Figure E-14. (Continued)

```
      RETURN
      END
      SUBROUTINE SMAPLL(PHI,LPCTR,PHIREF,CONST,RATE,SUMDIF)
C     ROUTINE PERFORMS DIGITAL FILTERING ON INCOMING OMEGA PHASE
C     MEASUREMENTS( PLL, BURST FILTER).  OPERATION AND TIME CONSTANTS
C     CLOSELY RESEMBLE HARDWARE MAPLL.
C     ''CONST'' IS A VALUE TO BE SPECIFIED FOR MAX. LOOP CTR. VALUE
C     BEFORE ADJUSTING PHIREF. (LOOP TIME CONSTANT OF AMOUNT OF INTEGRA.
C
C     R. SALTER, AVIONICS ENGINEERING CENTER, JULY 1976.
C
      INTEGER PHI,PHIREF,CONST,RATE,LPCTR
      INTEGER DIFF,SUMDIF
C     COMPARE INCOMING PHASE TO LOOP CONTROL WORD(LCW) (PHASE REFERENCE
      DIFF=PHI-PHIREF
C     KEEP A RUNNING SUM OF ABSOLUTE VALUES OF DIFFERENCES TO INDICATE
C     SIGNAL VARIANCE.
      IF(DIFF) 10,50,20
C     CHECK FOR EDGE OF LANE AMBIGUITY.
   10 IF(DIFF.LT.-128)GO TO 25
      SUMDIF=SUMDIF+IABS(DIFF)
      GO TO 40
   20 IF(DIFF.GT.128)GO TO 35
      SUMDIF=SUMDIF+IABS(DIFF)
      GO TO 30
C     INCREMENT LOOP COUNTER.
   30 LPCTR=LPCTR+1
C     EXAMINE LPCTR FOR INCR. MORE THAN 'CONST' TIMES CORRELATED.
      IF(LPCTR.LE.CONST) RETURN
      LPCTR=0
      PHIREF=PHIREF+1
      RATE=RATE+1
C     CHECK FOR MODULO 255 LANE COUNTS.
      IF(PHIREF.GT.255) PHIREF=0
      RETURN
C     DECREMENT LOOP COUNTER.
   40 LPCTR=LPCTR-1
C     EXAMINE LPCTR FOR DECR. MORE THAN 'CONST' TIMES CORRELATED.
      IF(LPCTR.GE.-CONST) RETURN
      LPCTR=0
      PHIREF=PHIREF-1
      RATE=RATE-1
C     CHECK FOR MODULO 255 LANE COUNTS.
      IF(PHIREF.LT.0) PHIREF=255
   50 RETURN
   25 PHI=PHI+256
      DIFF=PHI-PHIREF
      SUMDIF=SUMDIF+IABS(DIFF)
      GO TO 30
   35 PHI=PHI-256
      DIFF=PHI-PHIREF
      SUMDIF=SUMDIF+IABS(DIFF)
      GO TO 40
      END
      SUBROUTINE PLOT1(X,LL)
```

ORIGINAL PAGE IS OF POOR QUALITY

Figure E-14. (Continued)

```
C      ROUTINE PLOTS DATA VECTOR "X" WITH STARS, ONE POINT PER LINE ON A
C      1-128 SCALE.
C
C      IF VALUES OF "X" CAN FALL BETWEEN 128-255, SET LL=1 AND ROUTINE
C      WILL HALVE EACH VALUE BEFORE PLOTTING.  OTHERWISE, IF VALUES OF
C      "X" ARE ALL LESS THAN 128, ROUTINE WILL PLOT STRAIGHT VALUES IF
C      LL=0
C
C      MUST DIMENSION X = VECTOR OF VALUES TO BE PLOTTED.
C      R.W. LILLEY, AVIONICS, JANUARY,1976
C      MODIFIED BY R. SALTER, AVIONICS, APRIL,1976.
C
      INTEGER BK/' '/,ST/'*'/,LIN(128)
      INTEGER X(1000)
      DO 10 I=1,128
   10 LIN(I)=BK
      DO 20 I=1,200
      IF(LL.EQ.0)GO TO 12
      MM=X(I)/2+1
      GO TO 13
   12 CONTINUE
      MM=X(I)
   13 CONTINUE
      LIN(MM)=ST
      PRINT 11,LIN
   11 FORMAT(1X,128A1)
      LIN(MM)=BK
   20 CONTINUE
      RETURN
      END
```

Figure E-14. (Continued)

```
C       SMAPLL SIMULATION ANALYSIS TO OPTIMIZE LOOP PARAMETERS
C       R. SALTER ; JANUARY 1977 ; AVIONICS ENGINEERING CENTER.
        REAL GA(5000),GB(5000),G(5000),INTVL,XX(100),YY(100)
        REAL S(600),STD(600)
        INTEGER NN(5000),ERROR(200)
        INTEGER NOISE(5000),TME,PHI(5000),PHIIN,PHIOUT ,SUMDIF,CTR2
        INTEGER ERRSUM,TSLOT,PHASE,DELTA,CONST,RATE,HALFLN
        DIMENSION YYY(101),XXX(101),XXXX(101),YYYY(101)
        REAL XXXL(12),YYYL(12),XXXXL(12),YYYYL(12),XXL(12),YYL(12)
        DATA XXL/'S','I','G','N','A','L','/','N','O','I','S','E'/
        DATA YYL/'P','L','L',' ','S','T','D',' ','D','E','V',' '/
C
C       XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
C
C       INITIALIZE PARAMETERS: A=SIGNAL APLITUDE; LEVELS =NUMBER
C       LOOP QUANTIZING LEVELS; CONST=NUMBER BITS SKIPPPED IN LOOP
C       BIDIRECTIONAL COUNTER(LOOP FILTER); INTVL=SAMPLING RATE IN
C       SECONDS; VEL=VELOCITY OF PHASE RAMP INPUT TO LOOP; PHASE=
C       INITIAL PHASE INPUT; SOGAIN=SECOND ORDER GAIN.
C       CONTRL IS A CONTROL WORD TO SPECIFY OPTIONAL CALCULATIONS AND PLOT
        CONTRL=0.
        LL=1
        A=3.
        SOGAIN=.00
        VEL1=300.
        VEL=VEL1
        PRINT 113,SOGAIN,VEL
  113   FORMAT(1X,'SECOND ORDER GAIN = ',F5.2,10X,'PHASE VELOCITY =',F7.2)
        HALFLN=64
        LANE=127
        LEVELS=128
        TSLOT=100
        NUM=5000/TSLOT
        CONST=1
        SMALLA=189.
        INTVL=.01
        TME=5000
        CAPA=2.*SQRT(4./SMALLA)
C
C       XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
C
C       GENERATE NOISE VIA CALLS TO RANDOM NUMBER GENERATORS AND
C       LOW PASS FILTERS TO CORRELATE IT.
        CALL INTGEN(5**15)
        CALL GRANGE(GA,INTVL,CAPA,SMALLA,TME)
C       COMPUTE THE MEAN AND VARIANCE OF NUMBERS RETURNED FROM RANDNR.
C       AND THROUGH RC LOW PASS FILTER GRANGE.
        CALL MNVAR(GA,AVE,VARI)
        CALL BNTGEN(5**11)
        CALL GRANGE(GB,INTVL,CAPA,SMALLA,TME)
    9   FORMAT(1X,10E10.2)
        SNR=10.*ALOG10(A*A/(VARI*2.))
C
C       XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
C
```

Figure E-15. FORTRAN Program RSSIM Used to Simulate SMAPLL2 and Evaluate Performance.

```
C       PERFORM CALCULATION OF LOOP VARIANCE FOR SEVERAL SNRS
C       SEVERAL TIMES. EACH TIME INCREASING THE LOOP TIME CONSTANT
C       (M)  (MORE INTEGRATION OR SKIIPED BIITS IN THE COUNTER).
        DO 45 MM=1,8
        PRINT 46,SOGAIN
   46   FORMAT(1X,'SECOND ORDER GAIN = ',F5.2)
        DO 40 M=1,1
        CONST=16
        PRINT 11,CONST
   11   FORMAT(3X,'LOOP CONSTANT = ',I5)
C       PERFORM CALLS TO SMAPLL AND CALCULATE OUTPUT VARIANCE FOR
C       100 GIVEN SNRS . INCREASE SNR EACH AROUND THIS LOOP.
        DO 30 L=1,1
C       CALCULATE NEW NOISE PHASE INPUTS FOR EACH SNR.
    3   DO 10 I=1,5000
        G(I)=ATAN2(GR(I),(GA(I)+A))
        G(I)=G(I)*57.2957795
  114   FORMAT(1X,E9.1)
  112   FORMAT(1X,E12.3)
        NOISE(I)=INT(G(I))
   10   CONTINUE
        IF(CONTRL.LT.10.)GO TO 14
C       CALL HIST1 SUBROUTINE TO PLOT HISTOGRAM OF INPUT NOISE PHASE.
        CALL HIST1(NOISE,5000,SNR,VAR)
   14   CONTINUE
        K=TSLOT
        ERPSUM=0
        SUMDIF=0
        CTR2=0
C       CALCULATE RATE FOR 2ND ORDER LOOP STEADY STATE TRACKING FOR GIVEN
C       VELOCITY SIGNAL INPUT.  (I.E. START LOOP OUT BEING LOCKED IN SS).
        IF(SOGAIN.EQ.0.)GO TO 17
        RATE=INT(VEL*FLOAT(LEVELS)/(5760.*SOGAIN))
   17   CONTINUE
        PHIOUT=0
        RATE=0
        LPCTR=0
        JJ=1
        J=0
        VEL=VEL1
        DELTA=INT(VEL/16.)
        PHASE=0
C       CALCULATE PHASE INPUT TO LOOP BY ADDING TRUE PHASE PLUS NOISE PHAS
   15   DO 20 I=1,K
        PHI(I+J)=PHASE
        PHIIN=PHI(I+J)+NOISE(I+J)
        PHIIN=MOD(PHIIN,360)
        IF(PHIIN.LT.0)PHIIN=PHIIN+360
        X=FLOAT(PHIIN)
        CALL QUANT(X,LEVELS)
        PHIIN=INT(X)
        PHIIN=MOD(PHIIN,LEVELS)
        IF(PHIIN.LT.0)PHIIN=PHIIN+LANE+1
        CALL SMAPLL(PHIIN,LPCTR,PHIOUT,CONST,CTR2,SUMDIF)
  111   FORMAT(1X,6I4)
```

Figure E-15. (Continued)

```
   20 CONTINUE
C
C      XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
C
C      CALCULATE LOOP ERROR BY TAKING PHI(TRUE PHASE IN) - PHIOUT
C      (LOOP ESTIMATE OF PHI).
       X=FLOAT(PHI(I+J))
       CALL QUANT(X,LEVELS)
       PHI(I+J)=INT(X)
       PHI(I+J)=MOD(PHI(I+J),LEVELS)
       IF(PHI(I+J).LT.0)PHI(I+J)=PHI(I+J)+LANE+1
       IQUANT=PHI(I+J)
       LQUANT=PHIOUT
       CONTRL=11.
       IF(CONTRL.LT.10.)GO TO 16
       CALL PLTPHS(LQUANT,IQUANT)
   16  CONTINUE
       CONTRL=0.
       ERROR(JJ)=PHIOUT-PHI(I+J)
       IF(ERROR(JJ).GT.HALFLN)ERROR(JJ)=ERROR(JJ)-LEVELS
       IF(ERROR(JJ).LT.-HALFLN)ERROR(JJ)=ERROR(JJ)+LEVELS
       ERROR(JJ)=INT(FLOAT(ERROR(JJ))*5.625)
C
C      XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
C
C      PERFORM RATE-AIDING FOR 2ND ORDER SMAPLL.
       RATE=RATE+CTR2
       XA=FLOAT(PHIOUT)+FLOAT(RATE)*SOGAIN
       PHIOUT=INT(XA)
       PHIOUT=MOD(PHIOUT,LEVELS)
       IF(PHIOUT.LT.0)PHIOUT=PHIOUT+LEVELS
C      CALCULATE PHASE AT TIME OF NEXT TIME SLOT (10 SEC LATER).
       DELTA=INT(VEL/16.)
       PHASE=PHASE+DELTA
       PHASE=MOD(PHASE,360)
       IF(PHASE.LT.0)PHASE=PHASE+360
       J=J+TSLOT
       SUMDIF=0
       CTR2=0
       LPCTR=0
       JJ=JJ+1
       IF(J.GT.2500)VEL=-VEL1
       IF(J.LT.5000)GO TO 15
C
C      XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
C
C      PLOT HISTOGRAM OF LOOP OUTPUT ERRORS VS PHASE IN CEC.
       CONTRL=11.
       IF(CONTRL.LT.10.)GOTO 31
       CALL HIST1(ERROR,NUM,SNR,VAR)
   31  CONTINUE
       CONTRL=0.
       SNR=10.*ALOG10(A**2/(VARI*2.))
       XX(L)=SNR
       YY(L)=VAR
```

Figure E-15. (Continued)

```
      YY(L)=SQRT(VAR)
      S(LL)=SNR
      STD(LL)=YY(L)
      LL=LL+1
      A=A+0.1
   29 FORMAT(1X,'INPUT SNR = ',F7.3,10X,'OUTPUT VARIANCE = ',F8.3)
   30 CONTINUE
  999 NPTSPP=100
      NPLOTS=1
      GO TO 40
      CALL MMPLOT(XX,YY,NPTSPP,NPLOTS,XXL,YYL)
      A=0.1
   40 CONTINUE
      GO TO 44
   44 CONTINUE
      SOGAIN=SOGAIN+.125
   45 CONTINUE
      CALL MMPLOT(S,STD,100,6,XXL,YYL)
      STOP
  200 STOP
      END
C     XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
      SUBROUTINE QUANT(X,LEVELS)
C     ROUTINE QUANTIZES REAL X TO NUMBER OF LEVELS = LEVELS.
C     RETTURNS REAL X QUANTIZED.
      XX=FLOAT(LEVELS)
      X=(XX/360.)*X
      I=INT(X)
      X=FLOAT(I)
      RETURN
      END
C     XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
      SUBROUTINE GRANGE(Y,INTVL,A,SMALLA, TME)
      INTEGER TME
      REAL Y(5000), INTVL
      TAU=INTVL
      R=EXP(-SMALLA*TAU)
      PSI=(A*A*SMALLA)/4.
      CONS=SQRT(PSI*(1.-(R*R)))
      CALL ANORM(AN)
      Y(1)=AN*PSI
      DO 10 I=2,TME
      J=I-1
      CALL ANORM(AN)
    9 FORMAT(1X,10E10.2)
      Y(I)=AN*CONS+R*Y(J)
   10 CONTINUE
      RETURN
      END
C     XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
      SUBROUTINE SMAPLL(PHI,LPCTR,PHIREF,CONST,CTR2,SUMDIF)
C     ROUTINE PEFFORMS DIGITAL FILTERING ON INCOMING OMEGA PHASE
C     MEASUREMENTS( PLL, BURST FILTER).  OPERATION AND TIME CONSTANTS
C     CLOSELY RESEMBLE HARDWARE MAPLL.
C     ''CONST'' IS A VALUE TO BE SPECIFIED FOR MAX. LOOP CTR. VALUE
```

Figure E-15. (Continued)

```
C      BEFORE ADJUSTING PHIREF. (LOOP TIME CONSTANT OF AMOUNT OF INTEGRA.
CC
C      R. SALTER, AVIONICS ENGINEERING CENTER, JULY 1976.
C
       INTEGER PHI,PHIREF,CONST,CTR2,LPCTR
       INTEGER DIFF,SUMDIF,HALFLN,LANE
       HALFLN=64
       LANE=127
C      COMPARE INCOMING PHASE TO LOOP CONTROL WORD(LCW) (PHASE REFERENCE
       DIFF=PHI-PHIREF
C      KEEP A RUNNING SUN OF ABSOLUTE VALUES OF DIFFERENCES TO INDICATE
C      SIGNAL VARIANCE.
       IF(DIFF) 10,50,20
C      CHECK FOR EDGE OF LANE AMBIGUITY.
   10  IF(DIFF.LT.-HALFLN) GO TO 25
       SUMDIF=SUMDIF+IABS(DIFF)
       GO TO 40
   20  IF(DIFF.GT.HALFLN) GO TO 35
       SUMDIF=SUMDIF+IABS(DIFF)
       GO TO 30
C      INCREMENT LOOP COUNTER.
   30  LPCTR=LPCTR+1
C      EXAMINE LPCTR FOR INCR. MORE THAN 'CONST' TIMES CORRELATED.
       IF(LPCTR.LE.CONST) RETURN
       LPCTR=0
       PHIREF=PHIREF+1
       CTR2=CTR2+1
C      CHECK FOR MODULO 256 LANE COUNTS.
       IF(PHIREF.GT.LANE) PHIREF=0
       RETURN
C      DECREMENT LOOP COUNTER.
   40  LPCTR=LPCTR-1
C      EXAMINE LPCTR FOR DECR. MORE THAN 'CONST' TIMES CORRELATED.
       IF(LPCTR.GE.-CONST) RETURN
       LPCTR=0
       PHIREF=PHIREF-1
       CTR2=CTR2-1
C      CHECK FOR MODULO 256 LANE COUNTS.
       IF(PHIREF.LT.0) PHIREF=LANE
   50  RETURN
   25  PHI=PHI+LANE
       DIFF=PHI-PHIREF
       SUMDIF=SUMDIF+IABS(DIFF)
       GO TO 30
   35  PHI=PHI-LANE
       DIFF=PHI-PHIREF
       SUMDIF=SUMDIF+IABS(DIFF)
       GO TO 40
       END
C      XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
       SUBROUTINE PLOT1(X,LL)
C      ROUTINE PLOTS DATA VECTOR "X" WITH STARS, ONE POINT PER LINE ON A
C      1-128 SCALE.
C
C      IF VALUES OF "X" CAN FALL BETWEEN 128-255, SET LL=1 AND ROUTINE
```

Figure E-15. (Continued)

```
C       WILL HALVE EACH VALUE BEFORE PLOTTING.  OTHERWISE, IF VALUFS OF
C       "X" ARE ALL LESS THAN 128, ROUTINE WILL PLOT STRAIGHT VALUES IF
C       LL=0
C
C       MUST DIMENSION X = VECTOR OF VALUES TO BE PLOTTED.
C       R.W. LILLEY, AVIONICS, JANUARY,1976
C       MODIFIED BY R. SALTER, AVIONICS, APRIL,1976.
C
        INTEGER BK/' '/,ST/'*'/,LIN(128)
        INTEGER X(1000)
        DO 10 I=1,128
     10 LIN(I)=BK
        DO 20 I=1,200
        IF(LL.EQ.0)GO TO 12
        MM=X(I)/2+1
        GO TO 13
     12 CONTINUE
        MM=X(I)
     13 CONTINUE
        LIN(MM)=ST
        PRINT 11,LIN
     11 FORMAT(1X,128A1)
        LIN(MM)=BK
     20 CONTINUE
        RETURN
        END
C       XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
        SUBROUTINE GRANGB(Y,INTVL,A,SMALLA, TME)
        INTEGER TME
        REAL Y(5000), INTVL
        TAU=INTVL
        R=EXP(-SMALLA*TAU)
        PSI=(A*A*SMALLA)/4.
        CONS=SQRT(PSI*(1.-(R*R)))
        CALL BNORM(AN)
        Y(1)=AN*PSI
        DO 10 I=2,TME
        J=I-1
        CALL BNORM(AN)
        Y(I)=AN*CONS+R*Y(J)
     10 CONTINUE
        RETURN
        END
C       XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
        SUBROUTINE PLTPHS(LQUANT,IQUANT)
C PLTPHS PLOTS THE QUANTIZED PHASE VALUE OF THE INPUT PHASE (+) AND THE
C OUTPUT PHASE (*)
        INTEGER BLANK,STAR,PT,CROSS
        DATA BLANK/' '/,STAR/'*'/,PT/'-'/,CROSS/'+'/
        DIMENSION LINE(128)
        DO 10 K=1,128
10      LINE(K)=BLANK
        LINE(LQUANT+1)=STAR
C IF IT IS DESIRED THAT ONLY THE OUTPUT PHASE BE PLOTTED, SET
C LINE(IQUANT)= BLANK
```

Figure E-15. (Continued)

ORIGINAL PAGE IS
OF POOR QUALITY

```
      LINE(IQUANT+1)=CROSS
      IF(LQUANT.EQ.IQUANT) LINE(LQUANT+1)=PT
      PRINT 1,LINE
1     FORMAT(1X,128A1)
      RETURN
      END
C     XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
      SUBROUTINE MNVAR(GA,AVE,VAR)
      REAL GA(5000)
C     COMPUTE MEAN AND VARIANCE OF R.V.'S RETURNED FROM RANDOM NUMBER
C     GENERATOR AND LOW PASS FILTER.
      SUM=0.
      DO 5 I=1,5000
      SUM=SUM+GA(I)
    5 CONTINUE
      AVE=SUM/5000.
      SUM2=0.
      DO 6 I=1,5000
      SUM2=SUM2+(GA(I)-AVE)**2
    6 CONTINUE
      VAR=SUM2/5000
      PRINT 7,AVE,VAR
    7 FORMAT(1X,2E20.9)
      RETURN
      END
C     XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
      SUBROUTINE HIST1(NOISE,NUM,SNR,VAR2)
      INTEGER NOISE(5000),NN(5000)
      REAL YYY(101),XXX(101),XXXL(12),YYYL(12),XXL(12),YYL(12)

      DATA XXL/'S','I','G','N','A','L','/','N','O','I','S','E'/
      DATA YYL/'P','L','L',' ','S','T','D',' ','D','E','V',' '/
      DATA XXXL/'N','O','I' 'S','E',' ','P','H','A','S','E',' '/
      DATAYYYL/'H','I','S','T','O','G','R','A','M',' ',' ',' '/
C     COMPUTE AND PLOT PROBABILITY DENSITY OF THE INPUT NOISE BEFORE
C     IT IS ADDED TO THE SIGNAL PHASE, QUANTIZED, AND SENT TO SMAPLL.
      ICEC=-50
      DO 18 I=1,101
      YYY(I)=0
      DO 8 J=1,NUM
      NN(J)=INT(FLOAT(NOISE(J))/3.6)
      IF(NN(J).EQ.ICEC)YYY(I)=YYY(I)+1.
  307 FORMAT(1X,I5,F6.1)
    8 CONTINUE
      XXX(I)=FLOAT(ICEC)
      ICEC=ICEC+1
   18 CONTINUE
      DO 19 IK=1,101
      YYY(IK)=YYY(IK)/FLOAT(NUM)
   19 CONTINUE
      GO TO 20
      CALL MMPLOT(XXX,YYY,101,1,XXXL,YYYL)
   20 CONTINUE
      ISUM=0
      DO 115 I=1,NUM
```

Figure E-15. (Continued)

```
      ISUM=ISUM+NN(I)
  115 CONTINUE
      AVE=FLOAT(ISUM)/FLOAT(NUM)
      SUM=0
      DO 116 I=1,NUM
      ERR=FLOAT(NN(I))-AVE
      SUM=SUM+ERR**2
  116 CONTINUE
      VAR2=SUM/FLOAT(NUM)
      RETURN
      PRINT 117,AVE,VAR2,SNR
  117 FORMAT(1X,'MEAN=',F10.5,10X,'VARIANCE=',F10.5,10X,'SNR=',F10.5)
      RETURN
      END
```

Figure E-15. (Continued)

F. OSP Microcomputer Routines.

1. Main Routine. The "main" routine in this version of the OSP is simply an initialization and idling routine. This program initializes all the memory words needed by the OSP data processing subroutines and then simply waits to be interrupted by the microcomputer interface module with new Omega data, as flow charted in Figure F-1 and listed in Figure F-2. This main routine would be replaced by a more meaningful area navigation routine in a full-blown ONS. The navigation routine could perform navigation processing functions in the time between Omega data interrupts.

2. Interrupt Service Routine. When the main program is interrupted by the microcomputer interface module with new Omega data, the interrupt service routine is implemented as flow charted in Figure F-3 and listed in Figure F-4. The service routine first loads the new Omega data and updates a time-keeping register. It then checks a system status word to see if auto-sync is complete. If not, the data is passed to the auto-sync subroutine. If auto-sync is complete the time-keeping register is consulted to determine whether the current data sample is to be treated as Omega signal information or a transmission gap sample. If the sample falls in a time-slot the SMAPLL2 subroutine is called and the data used to update the estimate of phase for that time-slot. If the data is noise-gap information the time-keeping register is again consulted to determine if this is the first measurement taken during the current gap. If so, the data output subroutine is called and the rate-aiding for the second order SMAPLL2 is performed. Otherwise, the new data is simply forgotten and control is returned to the main routine to wait for the next interrupt.

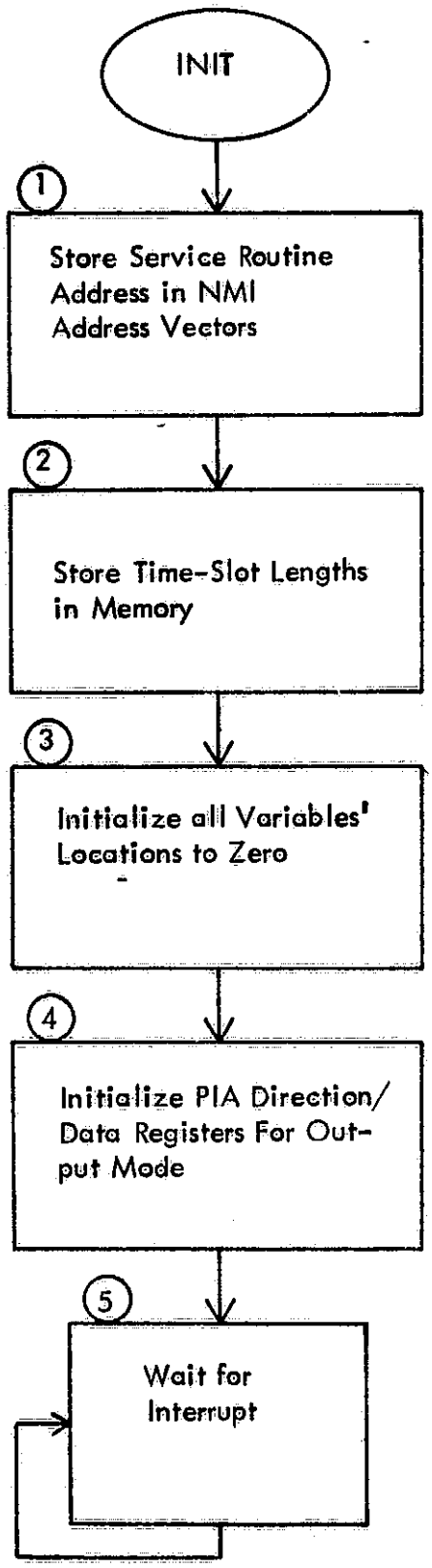


Figure F-1. Main (Initialization) Routine Flow Chart.

```
END PASS 1:   0 ERRORS

 1  0000                        ORG  00
 2  0000              NMIL      EQU  $FFFA
 3  0000              NMIH      EQU  $FFFB
 4  0000              OPTION    EQU  00
 5  0000              LOP1      EQU  01
 6  0000              LOP2      EQU  02
 7  0000              STAST     EQU  03
 8  0000              STAST2    EQU  74
 9  0000              LENGTH    EQU  04
10  0000              GAP       EQU  12
11  0000              TIMCTR    EQU  13
12  0000              SYNCST    EQU  15
13  0000              TEMP      EQU  16
14  0000              PHI       EQU  17
15  0000              PREVPH    EQU  18
16  0000              FIRST     EQU  19
17  0000              PTR1      EQU  20
18  0000              SHFCTR    EQU  21
19  0000              NSHIFT    EQU  23
20  0000              CORCTR    EQU  25
21  0000              MAXCOR    EQU  27
22  0000              SYNCPT    EQU  29
23  0000              CTR1      EQU  31
24  0000              CTR2      EQU  32
25  0000              WASTE     EQU  33
26  0000              LCW       EQU  34
27  0000              RATE      EQU  42
28  0000              SN        EQU  50
29  0000              TSLOT     EQU  66
30  0000              PTR3      EQU  67
31  0000              PTR4      EQU  68
32  0000              PTR2      EQU  69
33  0000              TEMP2     EQU  70
34  0000              MUXCTR    EQU  71
35  0000              BITTRN    EQU  90
36  0000              INPORT    EQU  $D000
37  0000              OUTPRT    EQU  $BC00
38  0000              HALFLN    EQU  63
39  0000              LANE      EQU  127
40  0000              THRESH    EQU  4
41  0000              NBYTE     EQU  125
42  0000              CONST     EQU  08
43  0000              LAF       EQU  90
44  0000              LBH       EQU  100
45  0000              LCE       EQU  110
46  0000              LDG       EQU  120
47  0000              LGAP      EQU  20
48  0000              SERVL     EQU  $60
49  0000              SERVH     EQU  $10
50  1000                        ORG  $1000
51  1000  78                    SEI
52  1001  A9 60       ONE       LDA  =SERVL
```

Figure F-2. Microcomputer Program INIT Listing.

```
 53   1003 8D FA FF          STA  NMIL
 54   1006 A9 10             LDA  =SERVH
 55   1008 8D FB FF          STA  NMIH
 56   100B A9 5A      TWO    LDA  =LAF
 57   100D 85 04             STA  LENGTH
 58   100F 85 09             STA  LENGTH+5
 59   1011 A9 64             LDA  =LBH
 60   1013 85 05             STA  LENGTH+1
 61   1015 85 0B             STA  LENGTH+7
 62   1017 A9 6E             LDA  =LCE
 63   1019 85 06             STA  LENGTH+2
 64   101B 85 08             STA  LENGTH+4
 65   101D A9 78             LDA  =LDG
 66   101F 85 0A             STA  LENGTH+6
 67   1021 85 07             STA  LENGTH+3
 68   1023 A9 14             LDA  =LGAP
 69   1025 85 0C             STA  GAP
 70   1027 A2 0D      THREE  LDX  =TIMCTR
 71   1029 A9 00             LDA  =00
 72   102B 95 00      FOUR   STA  OPTION,X
 73   102D E8                INX
 74   102E E0 4A             CPX  =STAST2
 75   1030 D0 F9             BNE  FOUR
 76   1032 A9 00      SIX    LDA  =00
 77   1034 85 0F             STA  SYNCST
 78   1036 A5 03             LDA  STAST
 79   1038 85 4A             STA  STAST2
 80   103A A9 00             LDA  =00
 81   103C 8D 01 5C          STA  $5C01
 82   103F A9 FF             LDA  =$FF
 83   1041 8D 00 5C          STA  $5C00
 84   1044 A9 04             LDA  =04
 85   1046 8D 01 5C          STA  $5C01
 86   1049 A9 01             LDA  =$01
 87   104B 85 42             STA  TSLOT
 88   104D 58                CLI
 89   104E 4C 4E 10   FIVE   JMP  FIVE
 90                          END

END PASS 2:    0 ERRORS
```

Figure F-2. (Continued)

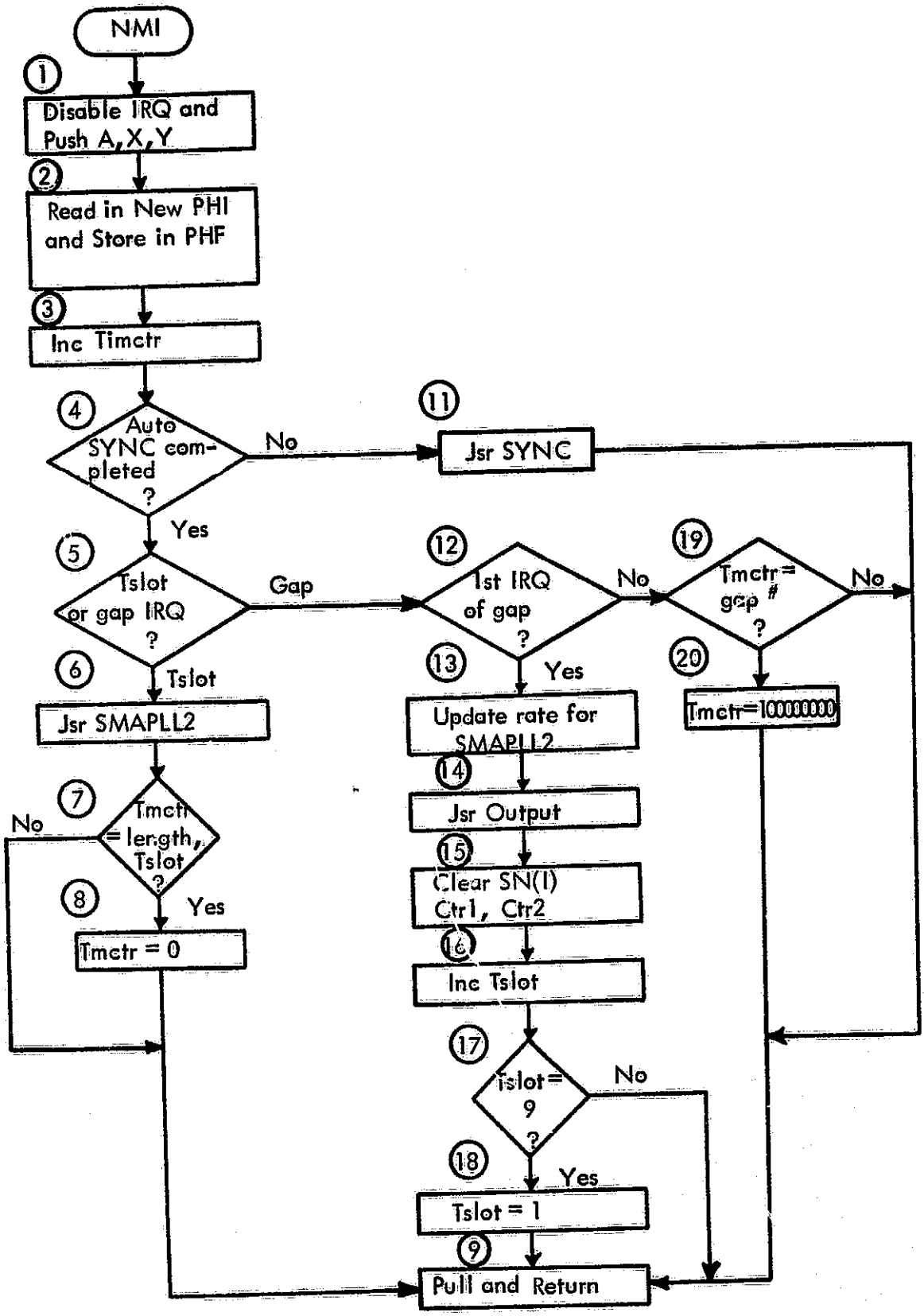


Figure F-3. Interrupt Service Routine Flow Chart.

```
END PASS 1:    0 ERRORS

 1  0000                        ORG 00
 2  0000             INPORT     EQU $D000
 3  0000             PHI        EQU 17
 4  0000             TIMCTR     EQU 13
 5  0000             SYNCST     EQU 15
 6  0000             TSLOT      EQU 66
 7  0000             LENGTH     EQU 04
 8  0000             PREVPH     EQU 18
 9  0000             CTR2       EQU 32
10  0000             RATE       EQU 42
11  0000             CTR1       EQU 31
12  0000             SN         EQU 50
13  0000             GAP        EQU 20
14  0000             SYNC       EQU $1100
15  0000             SMAPLL     EQU $1300
16  0000             OUTPUT     EQU $1400
17  1060                        ORG $1060
18  1060 78          ONE        SEI
19  1061 48                     PHA
20  1062 8A                     TXA
21  1063 48                     PHA
22  1064 98                     TYA
23  1065 48                     PHA
24  1066 AD 00 D0    TWO        LDA INPORT
25  1069 85 11                  STA PHI
26  106B E6 0E       THREE      INC TIMCTR+1
27  106D D0 02                  BNE FOUR
28  106F E6 0D                  INC TIMCTR
29  1071 A5 0F       FOUR       LDA SYNCST
30  1073 C9 03                  CMP =03
31  1075 D0 1D                  BNE ELEVEN
32  1077 A5 0E       FIVE       LDA TIMCTR+1
33  1079 10 31                  BPL TWELEV
34  107B 46 11       SIX        LSR PHI
35  107D 20 00 13               JSR SMAPLL
36  1080 A9 7F       SEVEN      LDA =%01111111
37  1082 25 0E                  AND TIMCTR+1
38  1084 A6 42                  LDX TSLOT
39  1086 D5 03                  CMP LENGTH-1,X
40  1088 D0 04                  BNE NINE
41  108A A9 00       EIGHT      LDA =00
42  108C 85 0E                  STA TIMCTR+1
43  108E 68          NINE       PLA
44  108F A8                     TAY
45  1090 68                     PLA
46  1091 AA                     TAX
47  1092 68                     PLA
48  1093 40                     RTI
49  1094 20 00 11    ELEVEN     JSR SYNC
50  1097 A5 0E                  LDA TIMCTR+1
51  1099 C9 E8                  CMP =$E8
52  109B D0 0C                  BNE B11
```

Figure F-4. Microcomputer Program OSERV Listing.

```
53   109D A5 0D          LDA TIMCTR
54   109F C9 03          CMP =$03
55   10A1 D0 06          BNE B11
56   10A3 A9 00          LDA =00
57   10A5 85 0E          STA TIMCTR+1
58   10A7 85 0D          STA TIMCTR
59   10A9 4C 8E 10  B11  JMP NINE
60   10AC C9 01     TWELEV CMP =01
61   10AE D0 3D          BNE NINETN
62   10B0 A6 42     THIRTN LDX TSLOT
63   10B2 A5 20          LDA CTR2
64   10B4 30 08          BMI LABEL1
65   10B6 18             CLC
66   10B7 75 29          ADC RATE-1,X
67   10B9 95 29          STA RATE-1,X
68   10BB 4C CC 10       JMP FOURTN
69   10BE 49 FF     LABEL1 EOR =$FF
70   10C0 18             CLC
71   10C1 69 01          ADC =01
72   10C3 85 20          STA CTR2
73   10C5 B5 29          LDA RATE-1,X
74   10C7 38             SEC
75   10C8 E5 20          SBC CTR2
76   10CA 95 29          STA RATE-1,X
77   10CC 20 00 14  FOURTN JSR OUTPUT
78   10CF 8A        FIFTN  TXA
79   10D0 18             CLC
80   10D1 65 42          ADC TSLOT
81   10D3 AA             TAX
82   10D4 A9 00          LDA =00
83   10D6 95 30          STA SN-2,X
84   10D8 95 31          STA SN-1,X
85   10DA 85 1F          STA CTR1
86   10DC 85 20          STA CTR2
87   10DE E6 42     SISTN  INC TSLOT
88   10E0 A9 09     SEVTN  LDA =09
89   10E2 C5 42          CMP TSLOT
90   10E4 D0 A8          BNE NINE
91   10E6 A9 01     EIGHTN LDA =01
92   10E8 85 42          STA TSLOT
93   10EA 4C 8E 10       JMP NINE
94   10ED C9 14     NINETN CMP =GAP
95   10EF D0 9D          BNE NINE
96   10F1 A9 80     TWENTY LDA =$80
97   10F3 85 0E          STA TIMCTR+1
98   10F5 4C 8E 10       JMP NINE
99                       END

END PASS 2:    0 ERRORS
```

Figure F-4. (Continued)

3. Automatic Synchronization Subroutine. The auto-sync procedure is explained in the main text, and the microcomputer program is flow charted in Figure F-5. The microcomputer language listing appears in Figure F-6.

4. SMAPLL2 Tracking Subroutine. The memory-aided second order phase locked loop (SMAPLL2) operation is also functionally outlined in the text. More details on the performance which can be expected from this tracking configuration is given in Appendix E. The SMAPLL2 procedure as implemented in the OSP microcomputer is charted in Figure F-7, and the program listing appears in Figure F-8.

5. Data Output Subroutine. The data output operation is described in the text and is flow charted in Figure F-9. The microcomputer output subroutine is listed in Figure F-10.

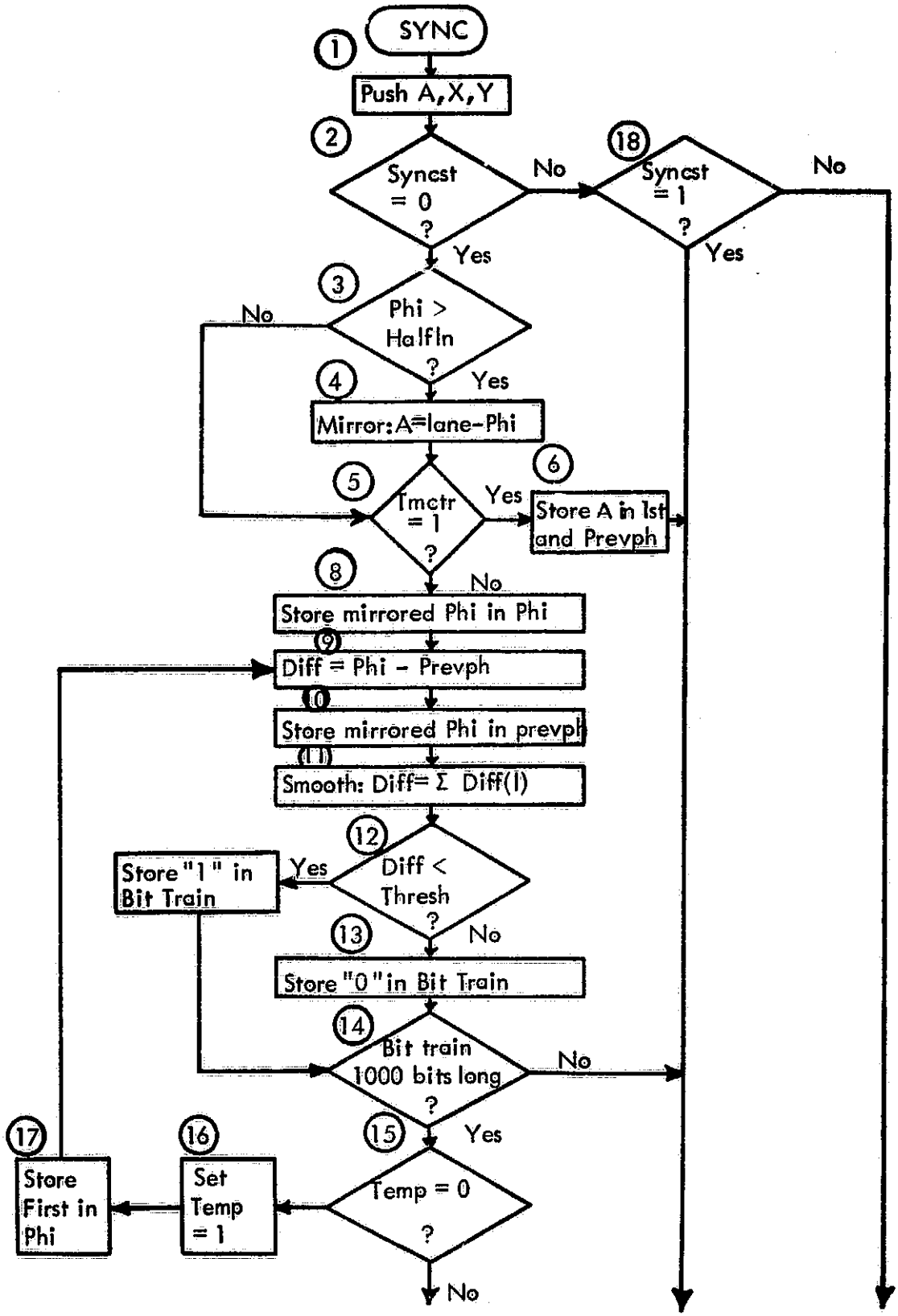


Figure F-5. Automatic Synchronization Subroutine Flow Chart.

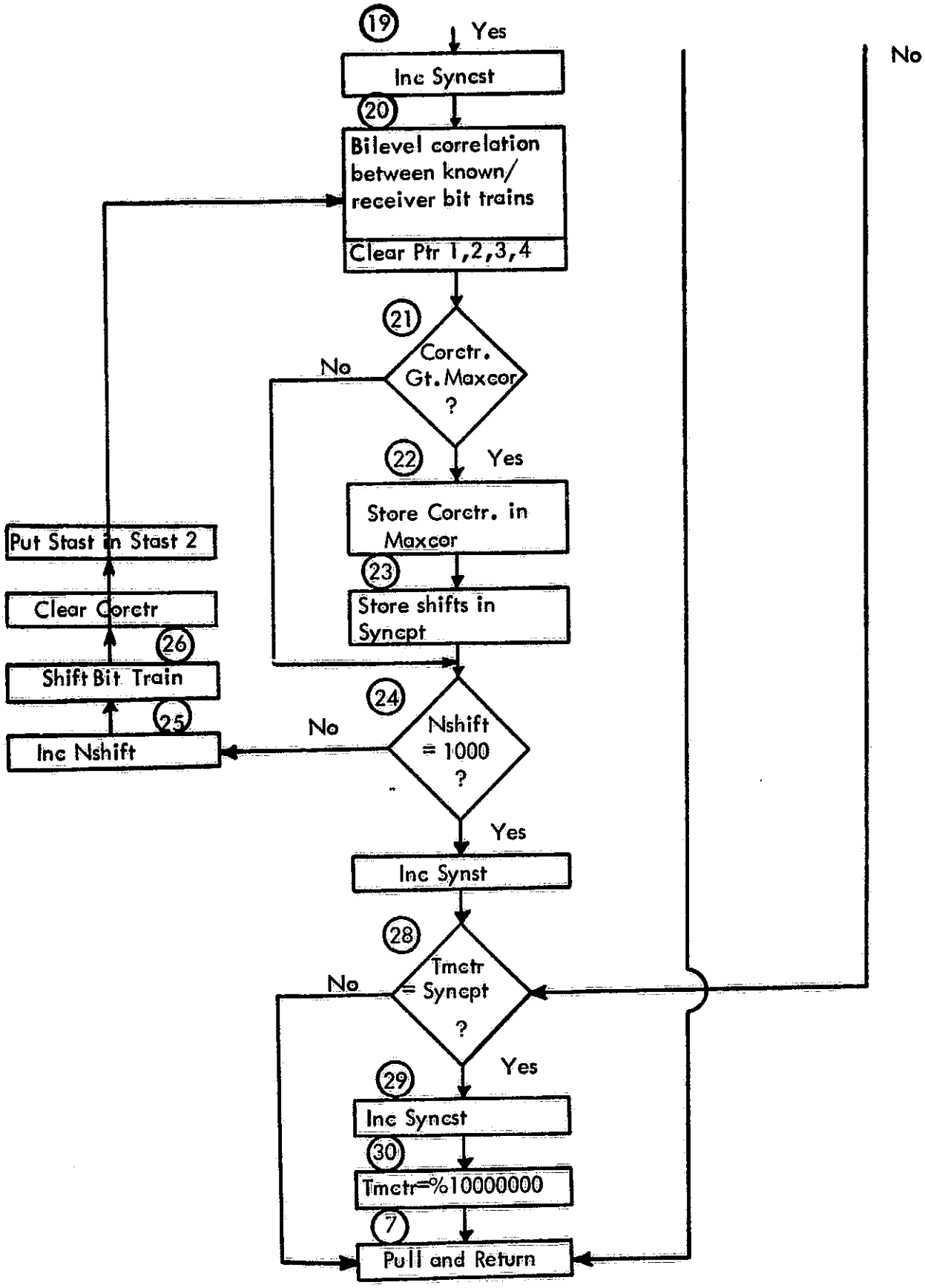


Figure F-5. (Continued)

```
END PASS 1:   0 ERRORS

 1  0000                        ORG  00
 2  0000              SYNCST    EQU  15
 3  0000              TEMP      EQU  16
 4  0000              PHI       EQU  17
 5  0000              SPHI      EQU  72
 6  0000              HALFLN    EQU  63
 7  0000              LANE      EQU  127
 8  0000              FIRST     EQU  19
 9  0000              PREVPH    EQU  18
10  0000              THRESH    EQU  05
11  0000              NBYTE     EQU  125
12  0000              BITTRN    EQU  90
13  0000              TIMCTR    EQU  13
14  0000              PTR2      EQU  20
15  0000              PTR1      EQU  69
16  0000              PTR3      EQU  67
17  0000              PTR4      EQU  68
18  0000              LENGTH    EQU  04
19  0000              STAST     EQU  03
20  0000              STAST2    EQU  74
21  0000              SHFCTR    EQU  21
22  0000              NSHIFT    EQU  23
23  0000              CORCTR    EQU  25
24  0000              MAXCOR    EQU  27
25  0000              SYNCPT    EQU  29
26  1100                        ORG  $1100
27  1100  48          SYNC      PHA
28  1101  98                    TYA
29  1102  48                    PHA
30  1103  8A                    TXA
31  1104  48                    PHA
32  1105  A5 0F       TWO       LDA  SYNCST
33  1107  D0 5B                 BNE  EIGHTN
34  1109  A5 11       THREE     LDA  PHI
35  110B  85 48                 STA  SPHI
36  110D  C9 3F                 CMP  =HALFLN
37  110F  90 05                 BCC  FIVE
38  1111  A9 7F       FOUR      LDA  =LANE
39  1113  38                    SEC
40  1114  E5 48                 SBC  SPHI
41  1116  A4 0E       FIVE      LDY  TIMCTR+1
42  1118  C0 01                 CPY  =01
43  111A  D0 0A                 BNE  EIGHT
44  111C  85 13       SIX       STA  FIRST
45  111E  85 12                 STA  PREVPH
46  1120  68          SEVEN     PLA
47  1121  A8                    TAY
48  1122  68                    PLA
49  1123  AA                    TAX
50  1124  68                    PLA
51  1125  60                    RTS
52  1126  85 48       EIGHT     STA  SPHI
```

Figure F-6. Microcomputer Program SYNC Listing.

```
53  1128 38          NINE   SEC
54  1129 E5 12             SBC  PREVPH
55  112B B0 05             BCS  TEN
56  112D 49 FF             EOR  =$FF
57  112F 18                CLC
58  1130 69 01             ADC  =01
59  1132 A4 48       TEN   LDY  SPHI
60  1134 84 12             STY  PREVPH
61  1136 EA          ELEVEN NOP
62  1137 EA                NOP
63  1138 EA                NOP
64  1139 C9 05       TWELEV CMP  =THRESH
65  113B 90 04             BCC  SIGNAL
66  113D 18          THIRTN CLC
67  113E 4C 42 11          JMP  STORE
68  1141 38          SIGNAL SEC
69  1142 A2 7D       STORE  LDX  =NBYTE
70  1144 36 59       ROTATE ROL  BITTRN-1,X
71  1146 CA                DEX
72  1147 D0 FE             BNE  ROTATE
73  1149 A5 0D       FOURTN LDA  TIMCTR
74  114B C9 03             CMP  =$03
75  114D D0 D1             BNE  SEVEN
76  114F A5 0E             LDA  TIMCTR+1
77  1151 C9 E8             CMP  =$E8
78  1153 D0 CB             BNE  SEVEN
79  1155 A5 10       FIFTN  LDA  TEMP
80  1157 D0 15             BNE  NINETN
81  1159 A9 01       SIXTN  LDA  =01
82  115B 85 10             STA  TEMP
83  115D A5 13       SEVTN  LDA  FIRST
84  115F 85 48             STA  SPHI
85  1161 4C 28 11          JMP  NINE
86  1164 C9 01       EIGHTN CMP  =01
87  1166 D0 03             BNE  A19
88  1168 4C 20 11          JMP  SEVEN
89  116B 4C 06 12    A19   JMP  TWENT8
90  116E E6 0F       NINETN INC  SYNCST
91  1170 A9 00             LDA  =00
92  1172 85 0E             STA  TIMCTR+1
93  1174 85 0D             STA  TIMCTR
94  1176 A6 14       TWENTA LDX  PTR2
95  1178 B5 04             LDA  LENGTH,X
96  117A 85 15       TWENTB STA  SHFCTR
97  117C 26 4A       TWENTE ROL  STAST2
98  117E 90 06             BCC  PLUS
99  1180 A9 01             LDA  =01
100 1182 85 10             STA  TEMP
101 1184 10 04             BPL  TWENTF
102 1186 A9 00       PLUS  LDA  =00
103 1188 85 10             STA  TEMP
104 118A A6 43       TWENTF LDX  PTR3
105 118C B5 5A               LDA  BITTRN,X
106 118E A4 44       TWENTG LDY  PTR4
107 1190 F0 04       B20G   BEQ  TWENTH
```

Figure F-6. (Continued)

```
108  1192 2A          ROL A
109  1193 88          DEY
110  1194 10 FA       BPL B20G
111  1196 2A     TWENTH ROL A
112  1197 A9 00  TWENTI LDA =00
113  1199 65 10       ADC TEMP
114  119B C9 01  TWENTJ CMP =01
115  119D F0 06       BEQ TWENTK
116  119F E6 1A       INC CORCTR+1
117  11A1 D0 02       BNE TWENTK
118  11A3 E6 19       INC CORCTR
119  11A5 A4 44  TWENTK LDY PTR4
120  11A7 C8          INY
121  11A8 C0 08  TWENTL CPY =08
122  11AA D0 04       BNE TWENTM
123  11AC A0 00       LDY =00
124  11AE E6 43       INC PTR3
125  11B0 84 44  TWENTM STY PTR4
126  11B2 C6 15       DEC SHFCTR
127  11B4 D0 D4  TWENTN BNE TWENTP
128  11B6 A5 45  TWENTO LDA PTR1
129  11B8 D0 08       BNE TWENTQ
130  11BA A9 14  TWENTP LDA =20
131  11BC 85 15       STA SHFCTR
132  11BE E6 45       INC PTR1
133  11C0 10 C4       BPL PLUS
134  11C2 A5 14  TWENTQ LDA PTR2
135  11C4 C9 07       CMP =07
136  11C6 F0 08       BEQ TWENT1
137  11C8 E6 14  TWENTT INC PTR2
138  11CA A9 00       LDA =00
139  11CC 85 45       STA PTR1
140  11CF F0 A6       BEQ TWENTA
141  11D0 A9 00  TWENT1 LDA =00
142  11D2 85 45       STA PTR1
143  11D4 85 14       STA PTR2
144  11D6 85 43       STA PTR3
145  11D8 85 44       STA PTR4
146  11DA A5 19       LDA CORCTR
147  11DC C5 1B       CMP MAXCOR
148  11DE 90 18       BCC TWENT4
149  11E0 D0 06       BNE TWENT2
150  11E2 A5 1A       LDA CORCTR+1
151  11E4 C5 1C       CMP MAXCOR+1
152  11E6 90 10       BCC TWENT4
153  11E8 A5 19  TWENT2 LDA CORCTR
154  11EA 85 1B       STA MAXCOR
155  11EC A5 1A       LDA CORCTR+1
156  11EE 85 1C       STA MAXCOR+1
157  11F0 A5 17  TWENT3 LDA NSHIFT
158  11F2 85 1D       STA SYNCPT
159  11F4 A5 18       LDA NSHIFT+1
160  11F6 85 1E       STA SYNCPT+1
161  11F8 A5 18  TWENT4 LDA NSHIFT+1
162  11FA C9 E8       CMP =$E8
```

Figure F-6. (Continued)

```
163  11FC D0 1D          BNE TWENT5
164  11FE A5 17          LDA NSHIFT
165  1200 C9 03          CMP =$03
166  1202 D0 17          BNE TWENT5
167  1204 E6 0F  TWENT7  INC SYNCST
168  1206 A5 0D  TWENT8  LDA TIMCTR
169  1208 C5 1D          CMP SYNCPT
170  120A D0 0C          BNE A25
171  120C A5 0E          LDA TIMCTR+1
172  120E C5 1E          CMP SYNCPT+1
173  1210 D0 06          BNE A25
174  1212 E6 0F  TWENT9  INC SYNCST
175  1214 A9 80  THIRTY  LDA =$80
176  1216 85 0E          STA TIMCTR+1
177  1218 4C 20 11 A25   JMP SEVEN
178  121B E6 18  TWENT5  INC NSHIFT+1
179  121D D0 02          BNE TWENT6
180  121F E6 17          INC NSHIFT
181  1221 A2 7D  TWENT6  LDX =NBYTE
182  1223 A5 5A          LDA BITTRN
183  1225 30 03          BMI MINUS
184  1227 18             CLC
185  1228 90 01          BCC DOIT
186  122A 38     MINUS   SEC
187  122B 36 59  DOIT    ROL BITTRN-1,X
188  122D CA             DEX
189  122E D0 FB          BNE DOIT
190  1230 A9 00          LDA =00
191  1232 85 19          STA CORCTR
192  1234 85 1A          STA CORCTR+1
193  1236 A5 03          LDA STAST
194  1238 85 4A          STA STAST2
195  123A 4C 76 11       JMP TWENTA
196                      END

END PASS 2:    0 ERRORS
```

Figure F-6. (Continued)

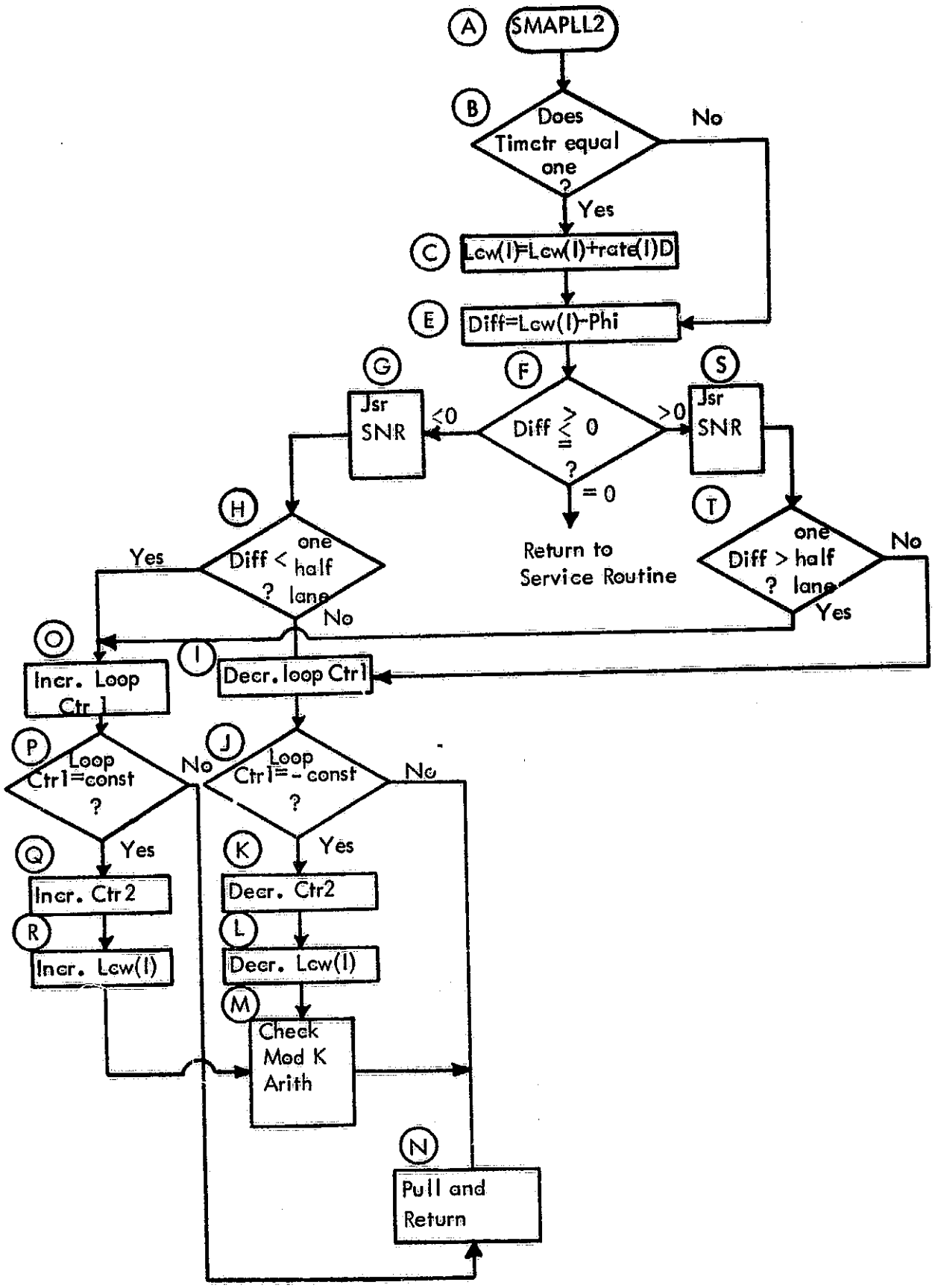


Figure F-7. SMAPLL2 Subroutine Flow Chart.

```
END PASS 1:    0 ERRORS

 1  0000                        ORG 00
 2  0000                TIMCTR  EQU 13
 3  0000                CTR1    EQU 31
 4  0000                CTR2    EQU 32
 5  0000                RATE    EQU 42
 6  0000                LCW     EQU 34
 7  0000                SN      EQU 50
 8  0000                PHI     EQU 17
 9  0000                TSLOT   EQU 66
10  0000                PREVPH  EQU 18
11  0000                TEMP    EQU 16
12  0000                TEMP2   EQU 70
13  0000                HALFLN  EQU 32
14  0000                CONST   EQU 08
15  0000                MOD64   EQU $1390
16  0000                SNR     EQU $13C0
17  1300                        ORG $1300
18  1300 48             SMAPLL  PHA
19  1301 8A                     TXA
20  1302 48                     PHA
21  1303 98                     TYA
22  1304 48                     PHA
23  1305 A5 0E          B1      LDA TIMCTR+1
24  1307 C9 81                  CMP =$81
25  1309 D0 25                  BNE E1
26  130B A6 42          C1      LDX TSLOT
27  130D B5 29                  LDA RATE-1,X
28  130F 30 0C                  BMI LABEL1
29  1311 4A                     LSR A
30  1312 4A                     LSR A
31  1313 85 46                  STA TEMP2
32  1315 B5 21                  LDA LCW-1,X
33  1317 18                     CLC
34  1318 65 46                  ADC TEMP2
35  131A 4C 2E 13               JMP D1
36  131D 49 FF          LABEL1  EOR =$FF
37  131F 18                     CLC
38  1320 69 01                  ADC =01
39  1322 4A                     LSR A
40  1323 4A                     LSR A
41  1324 85 46                  STA TEMP2
42  1326 B5 21                  LDA LCW-1,X
43  1328 38                     SEC
44  1329 E5 46                  SBC TEMP2
45  132B 20 90 13       D1      JSR MOD64
46  132E 95 21                  STA LCW-1,X
47  1330 A6 42          E1      LDX TSLOT
48  1332 B5 21                  LDA LCW-1,X
49  1334 38                     SEC
50  1335 E5 11                  SBC PHI
51  1337 F0 2A          F1      BEQ N1
52  1339 B0 41                  BCS S1
```

Figure F-8. Microcomputer Program SMAPLL2 Listing.

```
53   133B A5 11      H1      LDA PHI
54   133D 38                 SEC
55   133E F5 21              SBC LCW-1,X
56   1340 20 C0 13   G1      JSR SNR
57   1343 C9 20              CMP =HALFLN
58   1345 90 22              BCC O1
59   1347 C6 1F      I1      DEC CTR1
60   1349 A9 FF              LDA =255
61   134B 45 1F      J1      EOR CTR1
62   134D 18                 CLC
63   134E 69 01              ADC =01
64   1350 C9 08              CMP =CONST
65   1352 D0 0F              BNE N1
66   1354 A9 00              LDA =00
67   1356 85 1F              STA CTR1
68   1358 C6 20      K1      DEC CTR2
69   135A D6 21      L1      DEC LCW-1,X
70   135C B5 21      CKMOD   LDA LCW-1,X
71   135E 20 90 13   M1      JSR MOD64
72   1361 95 21              STA LCW-1,X
73   1363 68         N1      PLA
74   1364 A8                 TAY
75   1365 68                 PLA
76   1366 AA                 TAX
77   1367 68                 PLA
78   1368 60                 RTS
79   1369 E6 1F      O1      INC CTR1
80   136B A9 08      P1      LDA =CONST
81   136D C5 1F              CMP CTR1
82   136F D0 F2              BNE N1
83   1371 A9 00              LDA =00
84   1373 85 1F              STA CTR1
85   1375 E6 20      Q1      INC CTR2
86   1377 F6 21      R1      INC LCW-1,X
87   1379 4C 5C 13           JMP CKMOD
88   137C 20 C0 13   S1      JSR SNR
89   137F C9 20      T1      CMP =HALFLN
90   1381 B0 E6              BCS O1
91   1383 4C 47 13           JMP I1
92                           END

END PASS 2:    0 ERRORS
```

Figure F-8. (Continued)

```
END PASS 1:    0 ERRORS

   1  0000                          ORG  00
   2  0000                TEMP2     EQU  70
   3  1390                          ORG  $1390
   4  1390 85 46          MOD64     STA  TEMP2
   5  1392 8A                       TXA
   6  1393 48                       PHA
   7  1394 98                       TYA
   8  1395 48                       PHA
   9  1396 A5 46          TWO       LDA  TEMP2
  10  1398 30 11                    BMI  SEVEN
  11  139A 29 40          THREE     AND  =%01000000
  12  139C F0 06                    BEQ  SIX
  13  139E A5 46          FOUR      LDA  TEMP2
  14  13A0 29 3F                    AND  =%00111111
  15  13A2 85 46          FIVE      STA  TEMP2
  16  13A4 68             SIX       PLA
  17  13A5 A8                       TAY
  18  13A6 68                       PLA
  19  13A7 AA                       TAX
  20  13A8 A5 46                    LDA  TEMP2
  21  13AA 60                       RTS
  22  13AB 49 FF          SEVEN     EOR  =$FF
  23  13AD 85 46                    STA  TEMP2
  24  13AF E6 46                    INC  TEMP2
  25  13B1 A9 3F                    LDA  =63
  26  13B3 38                       SEC
  27  13B4 E5 46                    SBC  TEMP2
  28  13B6 4C A2 13                 JMP  FIVE
  29                                END

END PASS 2:    0 ERRORS
```

Figure F-8. (Continued)

```
END PASS 1:    0 ERRORS

 1  0000                          ORG  0
 2  0000                TEMP2     EQU  70
 3  0000                TSLOT     EQU  66
 4  0000                SN        EQU  50
 5  13C0                          ORG  $13C0
 6  13C0 85 46          SNR       STA  TEMP2
 7  13C2 8A                       TXA
 8  13C3 48                       PHA
 9  13C4 98                       TYA
10  13C5 48                       PHA
11  13C6 A5 42          TWO       LDA  TSLOT
12  13C8 18                       CLC
13  13C9 65 42                    ADC  TSLOT
14  13CB AA                       TAX
15  13CC A5 46                    LDA  TEMP2
16  13CE 18                       CLC
17  13CF 75 31                    ADC  SN-1,X
18  13D1 95 31                    STA  SN-1,X
19  13D3 90 03                    BCC  THREE
20  13D5 CA                       DEX
21  13D6 F6 31                    INC  SN-1,X
22  13D8 68             THREE     PLA
23  13D9 A8                       TAY
24  13DA 68                       PLA
25  13DB AA                       TAX
26  13DC A5 46                    LDA  TEMP2
27  13DE 60                       RTS
28                                END

END PASS 2:    0 ERRORS
```

Figure F-8. (Continued)

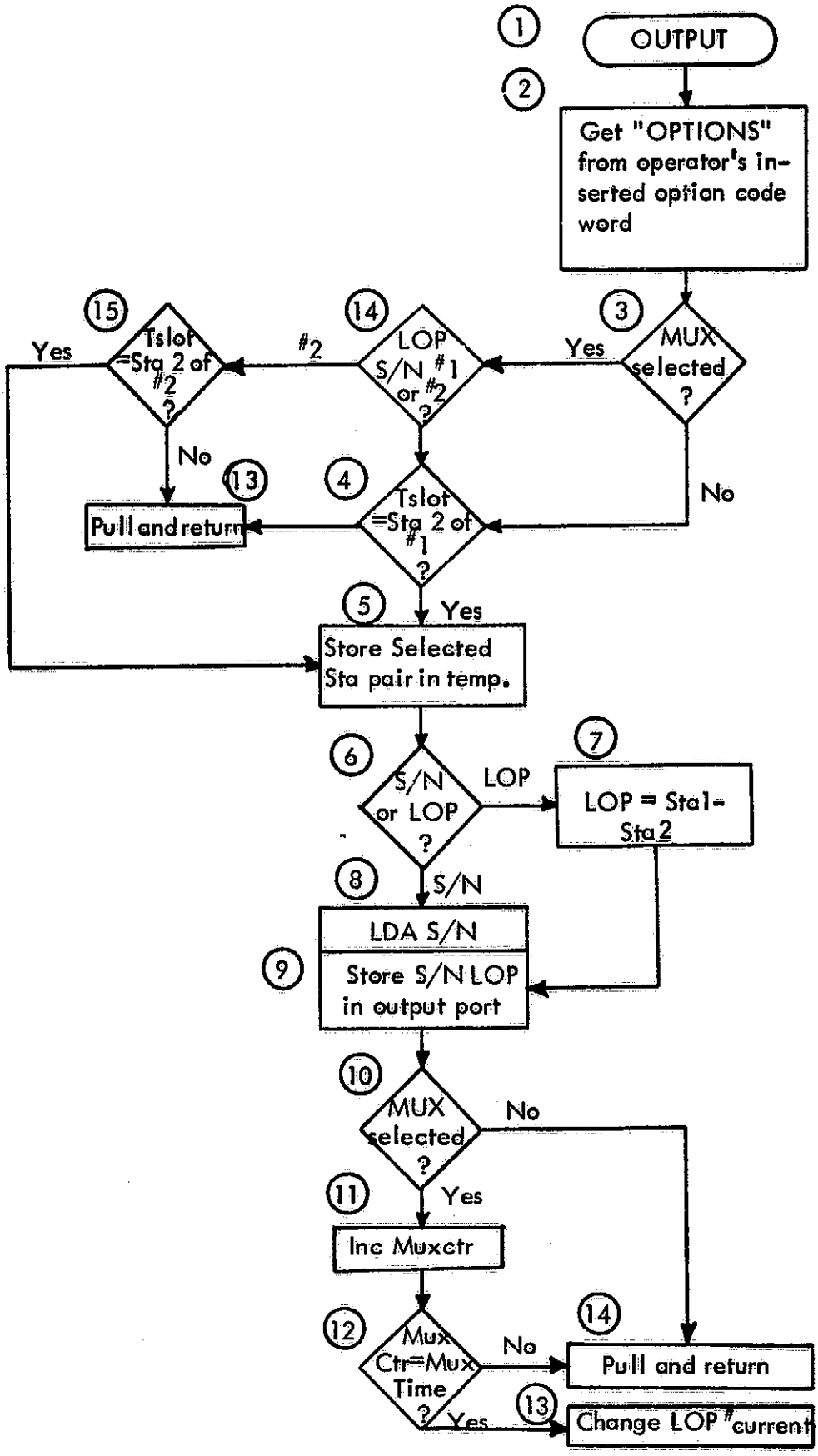


Figure F-9. Data OUTPUT Subroutine Flow Chart.

```
END PASS 1:   0 ERRORS

   1  0000                        ORG  00
   2  0000                OPTION  EQU  00
   3  0000                LOP1    EQU  01
   4  0000                LOP2    EQU  02
   5  0000                MUXCTR  EQU  71
   6  0000                WASTE   EQU  33
   7  0000                LCW     EQU  34
   8  0000                SN      EQU  50
   9  0000                TSLOT   EQU  66
  10  0000                TEMP    EQU  16
  11  0000                LANE    EQU  64
  12  0000                OUTPRT  EQU  $5C00
  13  1400                        ORG  $1400
  14  1400 48             OUTPUT  PHA
  15  1401 8A                     TXA
  16  1402 48                     PHA
  17  1403 98                     TYA
  18  1404 48                     PHA
  19  1405 A5 00          TWO     LDA  OPTION
  20  1407 30 6E          THREE   BMI  MUX
  21  1409 A5 01          FOUR    LDA  LOP1
  22  140B 85 10          FIVE    STA  TEMP
  23  140D 29 0F                  AND  =%00000111
  24  140F C5 42                  CMP  TSLOT
  25  1411 D0 5E                  BNE  FOURTN
  26  1413 A5 00          SIX     LDA  OPTION
  27  1415 29 40                  AND  =%01000000
  28  1417 F0 21                  BEQ  EIGHT
  29  1419 A5 10          SEVEN   LDA  TEMP
  30  141B 29 F0                  AND  =$F0
  31  141D 4A                     LSR  A
  32  141E 4A                     LSR  A
  33  141F 4A                     LSR  A
  34  1420 4A                     LSR  A
  35  1421 AA                     TAX
  36  1422 A5 10                  LDA  TEMP
  37  1424 29 0F                  AND  =$0F
  38  1426 85 10                  STA  TEMP
  39  1428 B5 21                  LDA  LCW-1,X
  40  142A A6 10                  LDX  TEMP
  41  142C 38                     SEC
  42  142D F5 21                  SBC  LCW-1,X
  43  142F 30 03                  BMI  COMPL
  44  1431 4C 51 14               JMP  NINE
  45  1434 18             COMPL   CLC
  46  1435 69 40                  ADC  =LANE
  47  1437 4C 51 14               JMP  NINE
  48  143A A5 42          EIGHT   LDA  TSLOT
  49  143C 18                     CLC
  50  143D 65 42                  ADC  TSLOT
  51  143F AA                     TAX
  52  1440 B5 30                  LDA  SN-2,X
```

Figure F-10. Microcomputer Program OUTPUT Listing.

```
53   1442 0A                ASL  A
54   1443 0A                ASL  A
55   1444 0A                ASL  A
56   1445 0A                ASL  A
57   1446 0A                ASL  A
58   1447 85 10             STA  TEMP
59   1449 B5 31             LDA  SN-1,X
60   144B 4A                LSR  A
61   144C 4A                LSR  A
62   144D 4A                LSR  A
63   144E 18                CLC
64   144F 65 10             ADC  TEMP
65   1451 8D 00 5C  NINE    STA  OUTPRT
66   1454 A5 00     TEN     LDA  OPTION
67   1456 10 19             BPL  FOURTN
68   1458 E6 47     ELEVEN  INC  MUXCTR
69   145A A5 47     TWELEV  LDA  MUXCTR
70   145C 29 3F             AND  =%00111111'
71   145E 85 10             STA  TEMP
72   1460 A5 00             LDA  OPTION
73   1462 29 3F             AND  =%00111111'
74   1464 C5 10             CMP  TEMP
75   1466 D0 09             BNE  FOURTN
76   1468 A5 47     THIRTN  LDA  MUXCTR
77   146A 29 C0             AND  =%11000000(
78   146C 18                CLC
79   146D 69 40             ADC  =%01000000(
80   146F 85 47             STA  MUXCTR
81   1471 68        FOURTN  PLA
82   1472 A8                TAY
83   1473 68                PLA
84   1474 AA                TAX
85   1475 68                PLA
86   1476 60                RTS
87   1477 A5 47     MUX     LDA  MUXCTR
88   1479 29 40             AND  =%01000000(
89   147B F0 8C             BEQ  FOUR
90   147D A5 02     FIFTEN  LDA  LOP2
91   147F 4C 0B 14          JMP  FIVE
92                         END

END PASS 2:    0 ERRORS
```

Figure F-10. (Continued)